SAMS

22310

The Tool Kit Series TI-99/4A Edition

Dave Dusthimer and Ted Buchholz

THE TOOL KIT SERIES
TI-99/4A

Ted Buchholz, a graduate of the University of South Carolina, is an editor with a publishing company in the Washington, D.C. area. He enjoys creative computer play with his children and their friends.

Dave Dusthimer is an avid home computer user and a self-taught programmer. As the father of three, Dave found himself teaching his children, as well as several neighborhood children, how to use and enjoy computers. A member of IEEE, Dave has edited and developed approximately 200 technical books over the past five years.

The Tool Kit Series

TI-99/4A Edition

by

Dave Dusthimer and Ted Buchholz

Howard W. Sams & Co., Inc.
4300 WEST 62ND ST. INDIANAPOLIS, INDIANA 46268 USA

FIRST EDITION
FIRST PRINTING—1984

International Standard Book Number: 0-672-22310-4
Library of Congress Catalog Card Number: 83-51669

Edited by: *Frank N. Speights*
Illustrated by: *Jill E. Martin* and *David K. Cripe*

Printed in the United States of America.

Preface

Thousands of home computers are collecting dust because their owners don't have the tools to make them work properly. All home appliances require tools to keep them humming along and your TI computer is no exception. In this book, you will find all the tools you need to make your microcomputer perform the way it should.

This book grew out of a need that we noted after buying our TI-99/4A. Most manuals and books approach BASIC programming by teaching statements, commands, and program data structures. This approach can be successful, but the average person is not motivated to learn program construction this way. After spending many frustrating evenings with our User's Manuals and our BASIC programming books, we still could not write the simple programs that we had in mind when we bought the computers. But finally, we discovered a way of learning to program that really works. By looking at programs in terms of their working parts—their subroutines—we were able to understand how to write simple programs. Soon we were designing our own games and quizzes. With the help of this book you can too.

The "tools" in this Tool Kit book are brief 5- to 15-line subroutines. Color, sound and music, graphics, animation, and computational subroutines are the "tools" and, when combined, form the basis for a variety of educational programs and computer games. Each line of each subroutine is carefully described and explained. We will tell you what each subroutine will do, how it can be changed, what the variables control, and hints for further experimentation. This approach makes programming easier, less frustrating, and a lot more fun. The modular form of programming is a sound method of structured programming (and it is acknowledged so in computer science) and it works well for the average microcomputer owner.

This book will also help parents teach their children how to program. Both of us have small children and the techniques used in this book have helped our children become computer literate. Building programs out of subroutines will give you, the user, quicker and more satisfying results that the traditional approaches. Since the subroutines are mostly educational and recreational, they will appeal to both parents and children.

In the Tool Kit books, we are sharing a valuable approach to building simple programs with the beginning computer user. Now, let's get to it and have some fun.

Dave Dusthimer
and Ted Buchholz

Dedication

I dedicate this book to my wife and children. To Joni, Michael and Christopher—thank you for being good while Dad sat and wrote. To Debbie, my wife—my thanks and gratitude for countless hours of typing and proofing. Your job was the toughest and I couldn't have done it without you. To all four of you, thanks for putting up with my growling and preoccupation.

D.D.

To my loving and supportive parents.

T.B.

Contents

CHAPTER 6

CHAPTER 7

CHAPTER 8

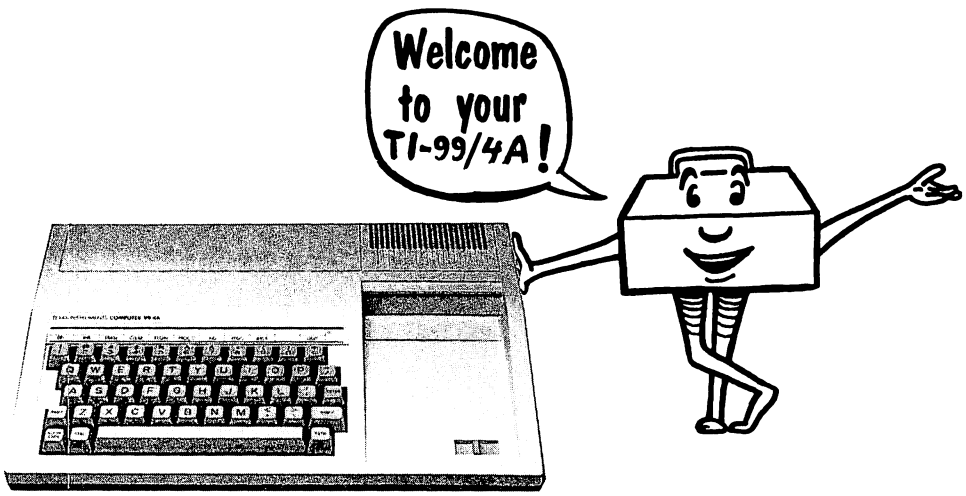
CHAPTER 9

CHAPTER 10

APPENDIX A

APPENDIX B

APPENDIX C



The TI-99/4A and How It Works 1

All computers are the result of genius level engineering. Fortunately for us "normal" people, you needn't be a genius to use one. Computers exist to serve you, the user. But since the TI microcomputer can't think like us, we must learn to think like it does. Don't worry, it isn't hard once you understand how the TI computer works (Fig. 1-1).

You have a TI-99/4A. Right? Well now you have the TI TOOL KIT.

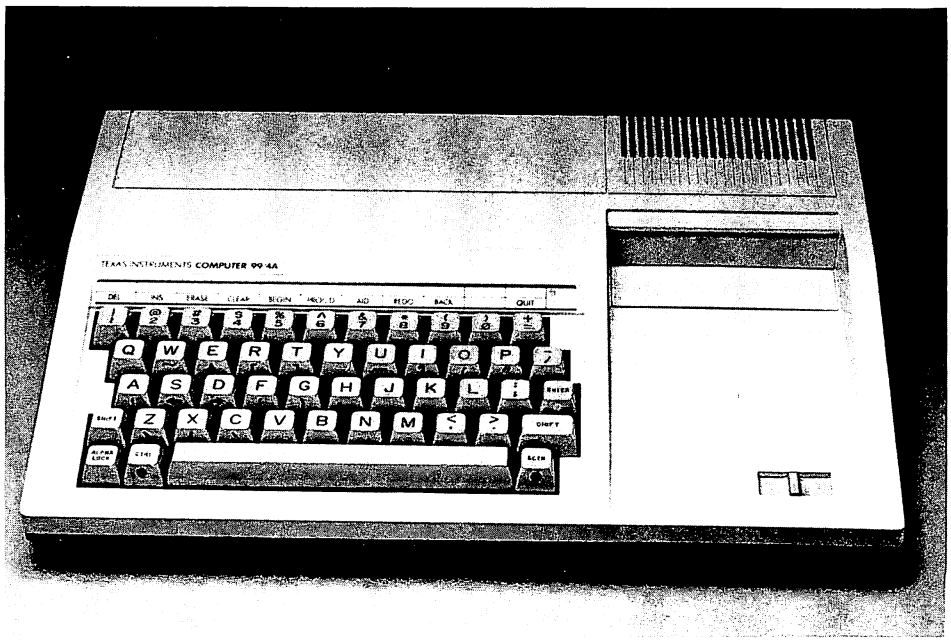


Fig. 1-1. The TI-99/4A microcomputer (Courtesy Texas Instruments Incorporated).

WHAT MAKES THE TI COMPUTER WORK?

At the core of every computer is its CPU, or Central Processing Unit. The CPU controls, interprets, and executes the computer functions. The TI's CPU is a TMS9900 microprocessor (Fig. 1-2). This microprocessor "chip" is very tiny, but it stores a great deal of information and is very powerful.

Computers can only receive information through their CPU in a binary system. That is, computers only know two things—whether data is

"on" or "off", "0" or "1", and "yes" or "no". Each of the digits (0 and 1) that are readable by the computer are known as *bits*. The TI-99/4A is called a 16-bit machine, because each of its characters is composed of 16 bits.

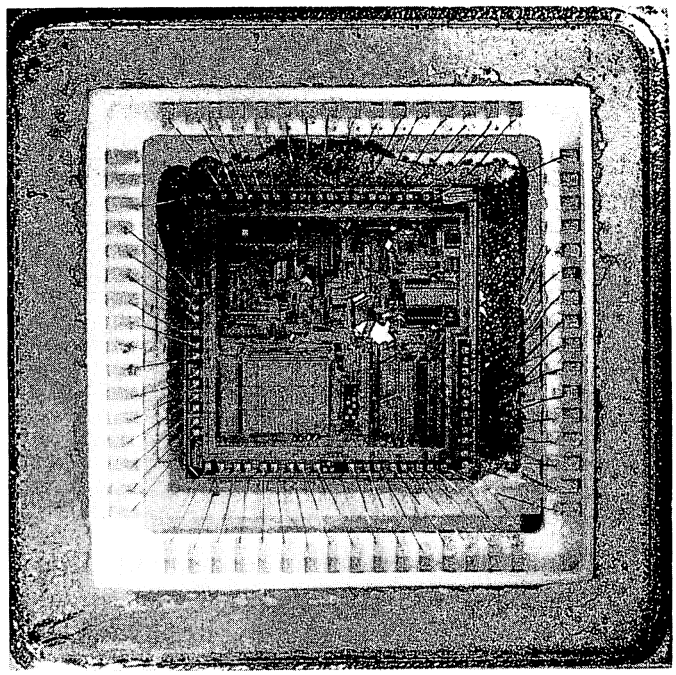


Fig. 1-2. The TMS9900 microprocessor (Courtesy Texas Instruments Incorporated).

We usually consider those bits that are put together to form a character or group as a *byte*. Knowing the number of bytes available in the microcomputer is another way of knowing just how much memory or space is available for your use.

HOW MUCH MEMORY DOES THE TI COMPUTER HAVE?

There are two types of memory in the TI-99/4A—read-only memory (ROM) and random-access memory (RAM). The ROM in the TI hard-

ware is programmed by the manufacturer and the CPU controls the computer operations by addressing programs located in ROM. The ROM is permanent in the computer and we, as users, cannot use the ROM for storage. The RAM (random-access memory) is a temporary storage memory that serves as space for the user's programs. The unexpanded TI machine (the standard TI computer without additional memory) has 16 kilobytes of RAM memory storage available.

You can expand the RAM by up to 48,000 individual memory locators (48,000 bytes or 48K bytes). The longer your programs, the more memory you will need. Unless you save the data in RAM (on tape or diskette), it will disappear when the computer is turned off. The standard 16K of memory is more than enough to handle all of the programs contained in this book.

THE BINARY SYSTEM

If the computer can only understand "0's" and "1's" in the binary system, how can you communicate with it?

Fortunately, there is an operating system that interprets the binary machine code into something more easily understood by humans. The operating system for the TI-99/4A is a BASIC interpreter, which allows us to communicate using TI BASIC as the programming language. Principally, we input data using the keyboard, but we can also use joysticks or other similar devices. Data are inputted using BASIC as a language so the machine can understand it.

HOW DO YOU WRITE PROGRAMS IN BASIC?

Programs are written with commands, statements, and a structure that is governed by rules not so different from grammar and syntax rules in English. We're lucky because BASIC is easier to learn than foreign languages. We will introduce you to the major commands and statements in the next chapter.

WHAT ACCESSORIES ARE NEEDED TO GET THE MOST OUT OF THE TI-99/4A?

You can do a lot with just the TI-99/4A and a screen (television or monitor). However, when you write programs of some length, you will

want to save yourself from constantly retyping (re-entering) the program every time you use it. For this purpose, you can use the TI Program Recorder (Fig. 1-3) to save those programs and files on tape, or you can purchase the TI expansion unit and a disk drive, and save your programs and files on floppy disks. The disk drive will turn your TI computer into a much more powerful machine because the drive can manage a lot of information without tapping the 16K of RAM in the TI-99/4A.

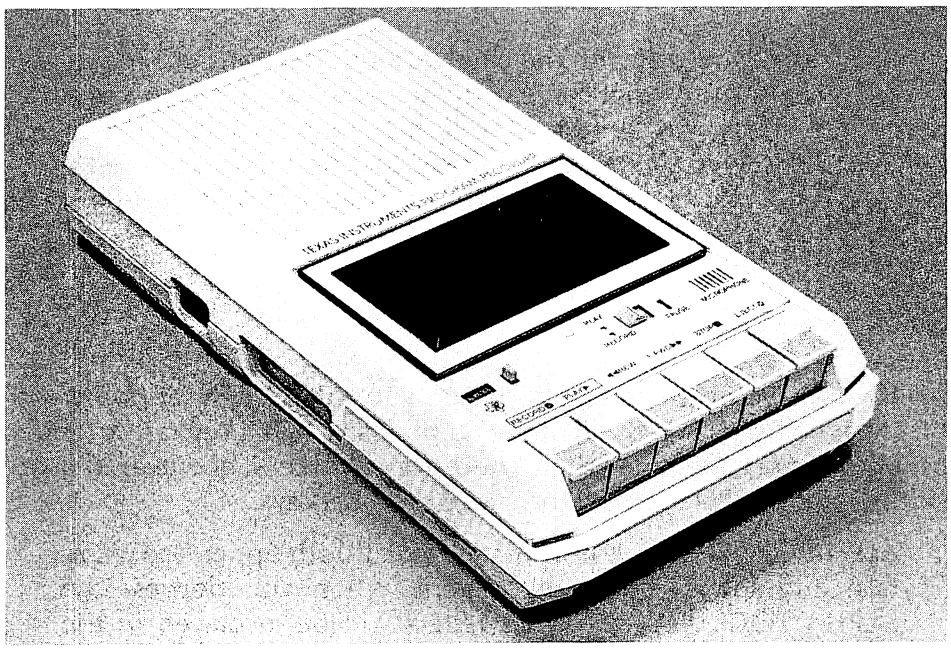


Fig. 1-3. The Texas Instruments Program Recorder (Courtesy Texas Instruments Incorporated).

An expanded version of BASIC is also available. TI Extended BASIC is a much more powerful language and it provides features not possible on standard TI BASIC. Extended BASIC allows you to use multiple statement lines which use less memory and are easier to input. If you do a lot of animation or graphics programming, you will want to consider adding Extended BASIC.

You can print out your programs or results using a printer; this will produce a "hard copy." There are many other accessories available at computer stores and through mail-order houses that will tailor the TI-

99/4A to your personal needs. You can add many extras to the TI computer as long as your supply of money holds out. But you don't have to spend a lot of money to enjoy the machine. We used only the TI computer, a cassette recorder, and a television set to do all of the programs in this book.

WHAT IS THE BEST WAY TO START LEARNING HOW TO USE THE TI-99/4A?

Familiarize yourself with the chief input device for the TI—the keyboard. With the guide that came with the computer, you can learn how to input characters and use the other functioning keys. Let's review this briefly and then you can use this section as a reference for the rest of the book. You cannot hurt the computer or destroy the ROM by playing around on the keyboard.

THE TI-99/4A KEYBOARD

If you have ever used a typewriter, you will be right at home with the TI keyboard (Fig. 1-4). All of the number and letter keys are arranged just like on a regular typewriter. There are some keys, however, that are a little different. Let's review the usage of these keys.

ALPHA LOCK—The TI-99/4A can display both uppercase (capitals) and lowercase (small) letters. When you depress the ALPHA LOCK key, all of the letters will be displayed in the uppercase mode.

FCTN—The function key is in the lower right-hand corner of the keyboard. When you press the function key, you activate the symbols printed on the front of the other keys. Quotation marks, question marks, and the brackets are some examples. To display these characters, press FCTN and the appropriate key. FCTN and the P key will display quotation marks.

You can also access special operations using the FCTN key. At the top of the keyboard, there is a plastic strip called the keyboard overlay which allows special keys to be identified. This two-level overlay strip looks like the diagram in Fig. 1-5. Its use is explained in detail in the General Information section of your User's Reference Guide.

Fig. 1-4. The keyboard (Courtesy Texas Instruments Incorporated).

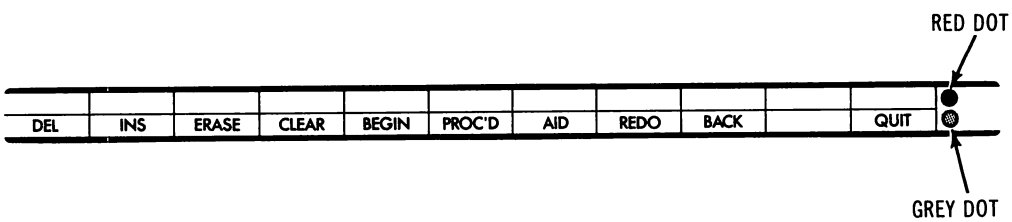


Fig. 1-5. The two-level strip overlay.

When you press any of the keys and the FCTN key at the same time, you access some time-saving editing devices.

DEL=FCTN+1—This deletes a character from the line you are typing. Move the cursor to the character to be deleted and hit FCTN+1. The character disappears.

INS=FCTN+2—This permits a character or space to be inserted in the line you are typing. Move the cursor to the desired

location, hit FCTN+2 and, then, type the character you want to insert.

ERASE=FCTN+3—This erases the line you are working on.

CLEAR=FCTN+4—This clears the screen of all characters.

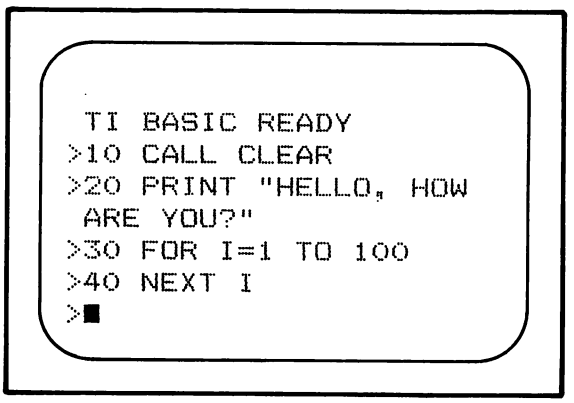
QUIT=FCTN+=SIGN—This key takes the TI-99/4A out of TI BASIC and returns it to the Color Bar Master screen.

BEGIN, PROC'D, AID, REDO, BACK—These keys have special software applications, but they are not used in this book.

The FCTN key also activates the Cursor Control arrows on the E, S, D, and X keys. You can move the cursor in the direction of the arrows by pressing the keys and FCTN at the same time.

INPUTTING PROCEDURES

If you have read your User's Manual, you already know that entering data is just like typing except that it is easier to make corrections on the TI computer. You should read your User's Manual if you haven't done so already. Here are some simple inputting rules and suggestions.



1. Always number your program lines. The TI-99/4A will do it for you if you input NUMBER and hit the ENTER key. If you number your own program lines, we suggest numbering in increments of at least 10. This gives you space in which to alter or change the program.

2. Get in the habit of using REM or remark statements as notes to yourself.
3. Be aware of punctuation marks. If you fail to use them properly, the programs won't run.
4. Watch out for spaces. Many commands require a space before more information is entered. GOSUB and GOTO are two examples.
5. Use the LIST command. As your programs get longer, you will need to know how to list certain segments of the program.

EDITING

Since very few of us are perfect, we are going to make inputting errors. The TI-99/4A makes it easy for us to fix mistakes.

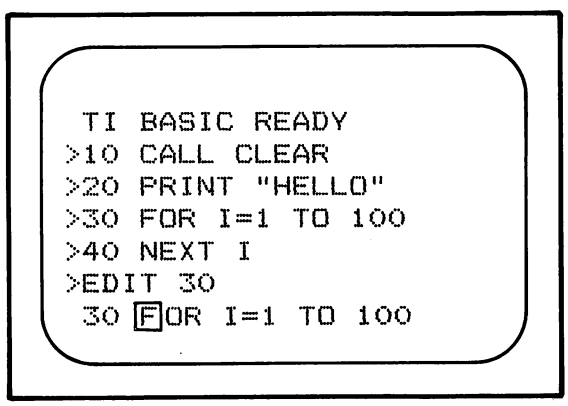
Editing Before the Line Is Entered

Using the Function key and the Cursor Controls, simply move the cursor to the mistake and type in the right character. You may have to correct other characters as well or you can use the Insert key. Now, let's make an intentional mistake and then correct it.

```
10 PRINT "MY TI 1000 IS A GREAT MACHINE"
```

Don't press ENTER. We have to change 1000 to -99/4A.

1. Using the Function key and the left Cursor Control key (the letter S), move the cursor to the space before 1 in 1000.



2. Now, type in -99/4A. Notice that we have used the space before the I in IS to make room for the extra character (space+1000 takes 5 spaces and -99/4A takes six spaces).
3. Hit the space bar and the space reappears, but the I in "IS" disappears.
4. You can now retype the rest of the line and hit ENTER.

You should now learn how to use the Insert key (Function 2). It will let you insert characters in the middle of a line without retyping the whole line.

Editing After the Line Is Entered

If you find a mistake after you have entered a line, or if you want to make a change in an earlier line, you would use the EDIT command. Let's change the TI-99/4A back to TI 1000. We will use our corrected line:

```
10 PRINT "MY TI-99/4A IS A GREAT MACHINE"
```

1. Input **EDIT 10.** Line 10 drops to the bottom of the screen and the cursor is flashing on the P in PRINT.
2. Using the Function key and the right Cursor Control key (letter D), move the cursor to the hyphen (-) just before the first 9 in 99. Hit the spacebar. This changes the hyphen to a space.
3. Now, type over the 99/4A with 1000 and hit the spacebar again to remove the A.
4. Now, hit ENTER and the correction will be made automatically. Type in LIST and see what happens.

The more you work with the TI-99/4A, the more mistakes you will make, so you will get plenty of editing practice.

THE TI-99/4A AS A CALCULATOR

Your TI computer is by its very nature a *number cruncher.* It can do addition, subtraction, multiplication, and division, as well as a variety of algebraic and trigonometric functions. The User's Manual describes in great detail how to do math with the TI-99/4A, so we will only do a quick review.

To use the TI computer as a calculator, all you need to do is use the PRINT statement and a number sentence. For addition, just type in:

```
PRINT 2 + 2 ENTER
4
```

and for subtraction,

```
PRINT 6 - 4 ENTER
2
```

Easy, isn't it? Notice that as soon as you hit the ENTER key, the answer appears on the next line on the screen. Now, for multiplication, type in

```
PRINT 5 * 2 ENTER
10
```

The TI-99/4A knows that it must multiply when it sees the asterisk, so you must always use the asterisk as the multiplication sign. For division, always use a virgule (a slash line):

```
PRINT 10 / 2
5
```

The TI-99/4A divides when it sees the slash line.

You can experiment with the math ability of the TI-99/4A as much as you like, but before you can perform problems that include different mathematical operations, you will need to know how the TI computer orders problems. Here is the *mathematical hierarchy* of the machine.

1. All expressions within parentheses are evaluated according to hierarchical rules.
2. Exponents are acted upon, in order, from left to right.
3. Prefix plus and minus are performed.
4. Multiplication and division are done.
5. Addition and subtraction are completed.

We will spend more time on mathematical operations later in Chapter 7.

VARIABLES

A variable is something that we assign a value to. We will assign values to variables in most of the programs in this book, so you will need to understand the concept.

If you say that X = 1, we are defining the variable X. If we put X = 1 in a program, every time the computer reads X, it will interpret X as a 1.

Variables can be used to help us perform mathematical operations. Our X = 1 is a *numeric* variable. We could use this variable in the following way:

```
10 LET X = 1
20 LET Y = 10 * X
30 PRINT X
40 PRINT Y
```

We have stated two variables, X and Y, in terms of each other; ten X's equal one Y.

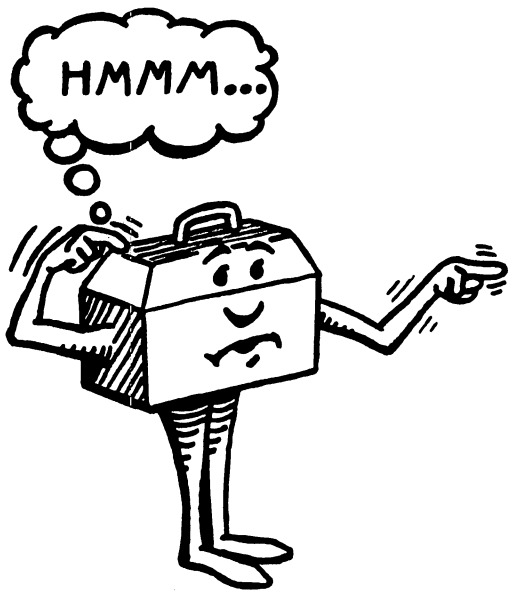
We can also state a variable in terms of a *string*. This type of variable is called, oddly enough, a string variable. If we want to represent a word with the letter X, we tell the computer to expect a string by adding a $ (dollar sign). Let's say we are designing a States and Capital game and we want a message to indicate an incorrect answer. We can say

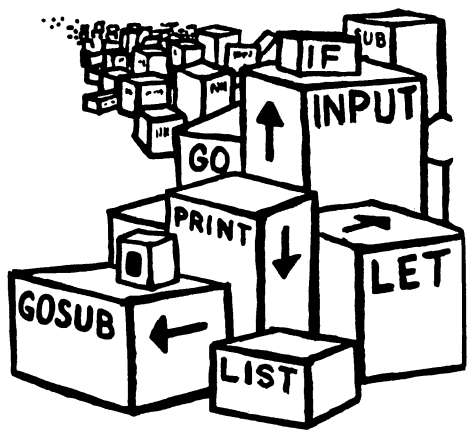
```
10 LET X$ = "WRONG"
20 PRINT X$
```

We are telling the computer that X$ means the same thing as the word WRONG.

As we begin working with programs in Chapter 3, you will see how valuable variables can be. They can state the value of nearly anything. We will identify the variables for you as we move on. But remember that there are two types of variables—NUMBER and STRING.

Now that we understand how the computer works and how to input data, let's learn how to speak its language—TI BASIC.





The Building Blocks of TI BASIC 2

Once you learn how the TI-99/4A operates and how to speak its language, learning to program will be easy. In Chapter 1, we explained how the TI computer works; now, we are going to teach you to speak TI BASIC. Remember that the TI-99/4A is a "dumb" machine. It can't do anything unless you, the user, command it to act. Before you get carried away with feelings of great power, you must also remember that the TI computer can't speak English (remember this from Chapter 1?), so you must give your commands in a language that the computer understands.

TI BASIC

Your TI-99/4A computer speaks TI BASIC. You can equip your TI to speak other languages, but this book deals only with Standard TI BASIC.

Your machine has a very limited vocabulary. It only understands about 70 commands, functions, or subprograms. In the next few pages, we will explain each command and will give you an example of how the command works. You will refer to this part of the book often, so you may want to put some notes in the margin. Now, let's learn how to talk TI BASIC.

COMMANDS AND STATEMENTS

The following commands are fairly standard BASIC commands. They can be used with most computers that understand BASIC.

BREAK—This command tells the TI computer to stop running a program or to take a "break" before it continues. When BREAK is entered, you must tell it at which line in the program to break. You will use this command very little.

BYE—When you enter this command, all open files are closed, all programs are erased, and you leave BASIC behind. When BYE is entered, the screen will display the first screen that it usually displays when the computer is turned on.

CLOSE—This statement simply closes a file that you opened earlier with an OPEN statement. You must always close a file before the TI-99/4A can go on to any other parts of a program.

CONTINUE—If you stop a program with a BREAK command,

you can use CONTINUE to begin the program again. You can also use **CON** instead of CONTINUE if you wish.

DATA—This command allows you to store information in either string or number form. You must tell the computer to read the DATA with a READ command. Data can be stored in a program by simply typing DATA followed by the information you want to store; i.e., **DATA, INFO,INFO,INFO,** and so on.

DEF—This statement allows you the opportunity to *define* your own functions for the needs of a specific program. You must enclose the parameters in parentheses right after the DEF statement.

DELETE—This command will only work if you have a disk drive connected to your TI-99/4A. This command will delete either a file or a program which you identify.

DIM—DIM lets you set the maximum size of a given array. For example, DIM(30) tells the TI-99/4A to reserve 30 elements for the array.

DISPLAY—This statement reacts just like the PRINT statement. It allows you to DISPLAY an output on the screen only. We rarely use this command.

EDIT—This command allows you to correct mistakes. In the immediate mode, just type EDIT and the line number you want to change. Like this:

```
EDIT 100
```

When you hit ENTER, the line will appear at the bottom of the screen. You can move the cursor using the Function key and correct the problem. When you finish correcting the line, hit ENTER again, and the corrected line will be stored in memory.

END—You guessed it. This statement stops or ends your program. You will use this statement most often to divide segments of a program or use at the end of a program. However, you don't have to use END at the end of your program. The TI computer will stop automatically after it reads the last line.

FOR/TO/STEP—Loops are easily created with this statement. The NEXT statement must always be used with a FOR/TO/STEP

or you will receive an error message. FOR and TO are used to state a variable. For example,

```
5 FOR A = 1 TO 5
```

STEP is used if you want the computer to "step" or cycle in increments greater than 1. If you want the computer to move in steps of 1, then you can leave the STEP portion of the command out. By using STEP and a negative number, you can create a negative step.

You must include the NEXT statement so that the computer will go back and act on the next variable in the FOR/TO/STEP command. Try this:

```
5  FOR I = 1 TO 5
10 PRINT "YOUR NAME"
15 NEXT I
```

GOSUB—This is just one of a couple of branching commands. You must use a RETURN statement to tell the computer to return to the main body of the program. The RETURN statement will send the computer to the line following the GOSUB statement.

GOTO—This is another branching statement. Following the GOTO statement and a SPACE, you must input the line number that you want the computer to read next. Once you use a GOTO statement, a continuous loop is made. To break the loop, another GOTO statement can be used to send the computer to another line in the program that is outside of the loop.

IF/THEN/ELSE—This is another branching statement but, this time, we are using a conditional branch. A condition described within the IF/THEN statement must occur before any branching can take place. For example:

```
IF X = 1 THEN 100
```

This line tells the computer that if the variable X = 1, then the computer is to go to line 100. If X doesn't equal one, then the computer will go to the next line and continue. When the ELSE statement is added, you can branch to another line. For example:

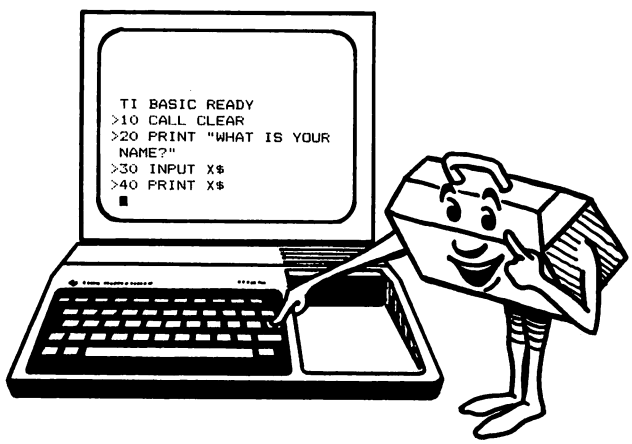
```
IF X = 1 THEN 100 ELSE 200
```

If X = 1, the computer goes to line 100; if X doesn't = 1, then the computer goes to line 200. This statement is very useful for constructing games and quizzes, and we will use it a great deal in later chapters.

INPUT—This statement halts the program and allows you to input a string or character using the keyboard.

Try this:

```
10 PRINT "WHAT IS YOUR NAME?"
20 INPUT X$
30 PRINT X$
```



LET—LET allows you to define variables in a program. You don't have to use the LET statement with the TI-99/4A, however. It interprets LET A = 1 and A = 1 the same way.

LIST—This command tells the computer to list the program statements in order. It can only be used in the immediate mode. You may also list specific lines of the program by entering the line number after the command. By typing a minus sign in front of the number, the computer will list all lines up to and including the number. Let's look at how the LIST function works.

LIST	-This lists the entire program.
LIST 400	-Only line 400 will appear.
LIST –400	-All lines up to and including 400 will be listed.
LIST 400–	-All lines including and after 400 will be listed.
LIST 200–400	-Lines 200 through 400 will be listed.

NEW—This command clears any program in memory from the memory. It acts just like an eraser. Be sure to save and store any programs you want to keep before entering NEW.

NEXT—This statement can only be used with a FOR/TO/STEP command. The NEXT statement tells the computer to evaluate the next value in the FOR/TO statement. As long as the value does not exceed the limits of the FOR/TO statement, the computer will act on the value. The NEXT statement controls the loop. If the values are within limits, the loop will continue. Once the values exceed the limits set in the FOR/TO/STEP statement, the NEXT statement will tell the computer to leave the loop and continue with the next line of the program.

NUMBER or **NUM**—This statement will automatically number your program lines for you, beginning with 100 and stepping in increments of 10. This makes programming much quicker. You can tell the computer what number to begin with and how many numbers to allow between lines if you like. For example, if you want the TI-99/4A to begin with 1000 and then list by 100's, you would enter **NUM 1000,100.** The first number tells the computer where to begin and the second number tells it how many numbers to leave between lines.

OLD—This command will give you a prompt sequence that allows you to load a program from a storage device (cassette tape or disk drive). If you have a compatible cassette recorder, you must add CS1 after the old command to get the prompts.

ON GOSUB—This is still another branching command. If a certain condition exists, the computer will go to the subroutine listed after GOSUB.

```
 5 INPUT X
10 ON X GOSUB 35,100,200
```

When value X is inputted, the computer will GOSUB line 35, 100, and 200. You must use a RETURN statement just like you do with GOSUB.

ON GOTO—This statement works just like the ON GOSUB statement except that you don't have to use the RETURN statement.

OPEN—OPEN tells a BASIC program to use files stored on memory storage devices. OPEN provides a necessary link between the storage device and a file number in the program.

OPTION BASE—This statement allows you to set the lower limits of an array at 1 instead of 0.

PRINT—When you want to display information on the screen, the PRINT statement will do it. Words to be printed must appear in quotation marks. When words, numbers, or strings follow the PRINT statement with the variables name, you don't need to use quotation marks.

RANDOMIZE—This statement along with the RND function will produce a sequence of numbers that appear to be random. This statement can be used with a "seed" number. In this way, the computer will begin randomizing with the same number each time. If a seed number is not used, the computer will begin at a different number each time.

READ—This statement tells the computer to read information from DATA statements in order, from left to right. You must use READ and DATA statements together.

REM—The computer does nothing with this statement. It is a notekeeping device for you, the user. REM or Remark statements are given line numbers and they identify parts of the program for user reference.

RESEQUENCE—After a program is debugged, this command will renumber the program lines. It will automatically redo GOTO and GOSUB numbers as well. You can resequence the

entire program or just a portion of it. You tell the computer what number to assign to the first line, and the increments to use between lines, by typing two sets of numbers after the RESEQUENCE command. It will look like this:

RES 100,10

This command will reorder the program. The first line number will be 100 and each following line will be ten places (values) greater.

100
110
120
ETC.

RESTORE—This statement tells the computer to go back to the beginning of a DATA statement and read the files again.

RETURN—This statement tells the computer to go back and read the line immediately following a GOSUB command. Each subroutine must end with a RETURN statement.

RUN—This is the computer's ignition key. When you command the computer to RUN, it will begin at the first line of the program and will carry out each program line in sequence.

SAVE—The SAVE command allows you to take a program from the computer's memory and store it on either a cassette tape or a floppy disk. If you are using a cassette recorder for memory storage, simply enter **SAVE CS1** and you will get a series of prompts that will lead you through the saving process.

STOP—This command is the same as END. It stops the program.

FUNCTIONS

Many of these functions will only work if you give the TI-99/4A a number to work with. The functions that require such a number are followed by the notation (X). The X is the argument.

ABS(X)—This function tells the computer to give you the absolute value of an expression. An *expression* is sometimes referred to as the *Argument*. The ABS command will always give

you a positive number even if the argument is negative. This command can be used to find out the absolute value of any complicated series of mathematical computations. For example, input:

```
ABS(47)
```

The absolute value is 47, but we know that. Now, try:

```
ABS 14 * (21 * 6.1) -20
```

Now do you see why this command is useful?

ASC(X)—Each character on the TI-99/4A has an ASCII code number. The ASC function will give you the code number for a string variable or for a group of numbers that appear in parentheses. Here's how it works.

```
 5 A$ = "D"
10 PRINT ASC(A$)
```

The number **68**, which is the ASCII code for the letter D, will appear on the screen.

ATN(X)—This function is rather mathematical and we won't use it in this book. This function will give you the arc tangent of any number that appears in parentheses after the ATN command.

**CHR$(X)**—This function is the exact opposite of the ASC function. The CHR$ function will change an ASCII code number into the appropriate character.

```
 5 A$ = CHR$(68)
10 PRINT A$
```

This will print the letter D.

COS(X)—This function will compute the cosine of any number that you put into parentheses, following the function. For example, enter:

```
COS(10)
```

EOF—If you use an OPEN statement to access files, you will also need to use the EOF (or END OF FILE) statement. EOF statements can also be used for branching. When an END OF FILE

condition exists, the GOTO or GOSUB commands tell the computer what to do.

EXP(X)—This returns the exponential function or the opposite of a logarithmic function. This function raises the number 2.718281828 to the power that you input in parentheses.

```
PRINT EXP(10)          -This will raise 2.718281828 to the
                        tenth power.
```

INT(X)—This function is used when you want a whole number as an output rather than a fraction or a decimal. The INT function returns the largest number possible that does not exceed the number in parentheses (the Argument). For negative numbers, the next integer value will be returned. Try this:

```
 5 A = INT(99.887)
10 PRINT A
```

The TI-99/4A will print 99. Try the same program only let A = INT(−99.887).

LEN(X)—This function will return the number of characters, including spaces in a string. Try this:

```
 5 A$ = "THE TI IS A NEAT MACHINE"
20 PRINT LEN(A$)
```

The TI-99/4A will print 24. Note that all the spaces count as a character.

LOG(X)—This function will give you the natural logarithm of a number. To figure the log of a number, simply input:

```
PRINT LOG(any number)
```

POS—This function will compare two strings and will tell you at what point a letter in one string begins occurring in another.

RND—The random function tells the TI-99/4A to generate the next random number that is controlled by the RANDOMIZE statement.

SEG$(X)—This function will give you a part of a string. You can control the part of the string returned.

```
 5 A$ = "MISSISSIPPI"
10  PRINT SEG$(A$,8,4)
```

The 8 in line 10 tells the TI-99/4A to count 8 places. The 4 tells the TI computer to print 4 places from the beginning point.

SGN(X)—This function will return the algebraic sign of the argument. It will tell you if the number is positive, negative, or zero. This function is useful in programs that require some mathematics.

```
10 PRINT SGN(82)
```

SIN(X)—The SIN function will return the sine of the argument.

SQR(X)—This handy function will give you the square root of any number you choose. Use the following statement to extract the square root.

```
PRINT SQR(Any number)
```

**STR$(X)**—STR$ is the opposite of the VAL function. STR$ changes a specified number (the argument) into a string.

TAB(X)—This function works just like the TAB key on a typewriter. With the PRINT statement, the TAB function tells the computer to begin printing a given number of places from the left side of the page. It looks like this:

```
PRINT TAB(20); "YOUR NAME"
```

The argument can be any number from 1 to 32.

TAN(X)—The TAN function will give you the tangent of the argument.

VAL(X)—This function will give you the numeric value for a string. The function ignores letters in a string and assigns numeric values of number characters only.

SUBPROGRAMS

The TI-99/4A understands several subprograms. These subprograms allow you to create sound, color, and graphics very easily. Subprograms are manufactured into the computer's memory and they allow you to call up colors, sounds, and graphic patterns. You specify what you want the computer to do by entering a series of numbers in parentheses following the subprogram. Let's look at the TI-99/4A subprograms.

CALL CLEAR—This subprogram clears the screen of the monitor. If you input this subprogram at the beginning of your programs, the screen will clear and the output will look better. You can also use CALL CLEAR any time that you want the screen to clear in the body of a program.

CALL CHAR—This allows you to create a new character and assign an ASCII number to it. You must name the character and tell the computer what the new character looks like. You describe the new character using a hexadecimal code that is represented by a 16-place string. A complete discussion of CALL CHAR can be found in Chapter 5. Here is the formula for CALL CHAR:

```
CALL CHAR(ANY ASCII number, "FFFFFFFFFFFFFFFF")
```

(NOTE: The F's can be replaced by other 16 place codes.)

CALL COLOR—CALL COLOR lets you change the colors that appear on the screen. This statement is always followed by numbers that represent a character-set number, foreground color code, and background color code. Here is the formula:

```
CALL COLOR(CHARACTER SET,FOREGROUND,
 BACKGROUND)
```

We will discuss CALL COLOR in detail in Chapter 3.

CALL GCHAR—CALL GCHAR will place a character anywhere on the screen. You must specify the row number, column number, and ASCII code for the character you want to display. The formula looks like this:

```
CALL GCHAR(ROW,COLUMN,ASCII CODE)
```

This subprogram is discussed in detail in Chapters 5 and 6.

CALL HCHAR—This subprogram will allow you to put a character anywhere on the screen and repeat it horizontally as many times as you wish. Here is the formula:

```
CALL HCHAR(ROW,COLUMN,CHARACTER,NUMBER OF
 REPEATS)
```

The number of repeats is optional. If you don't tell the TI-99/4A otherwise, it will display the character at the row and column

coordinates only once. We will discuss this command further in Chapters 5 and 6.

CALL JOYST—If you have joysticks, you can input data with them. This subprogram allows you to signal the computer to receive data from the joysticks.

CALL KEY—CALL KEY allows you to transfer a character directly into the program. This subprogram accepts the input from the keyboard and it can branch the program.

CALL SCREEN—When you use CALL SCREEN, you can change the screen color to any color in the TI palette. Here is the formula:

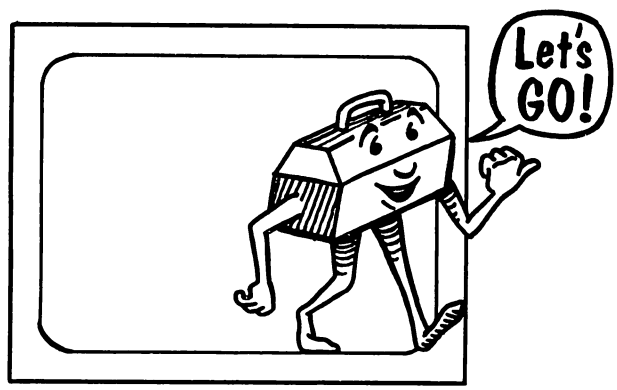
```
CALL SCREEN(# of color desired)
```

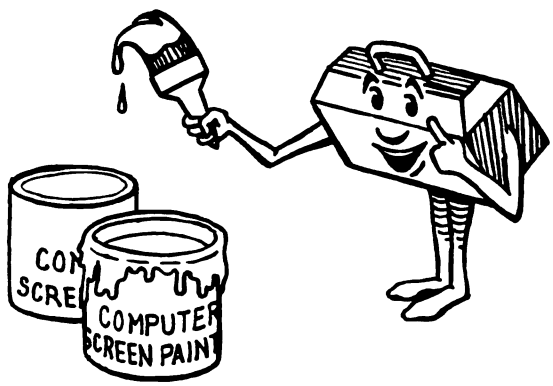
There is more discussion on this in Chapter 3.

CALL SOUND—This subprogram makes sounds, tones, or noise. You need only to input the duration of the tone, the frequency of the tone, and the volume you desire. We will discuss CALL SOUND at length in Chapter 4. Here is the formula:

```
CALL SOUND(DURATION,FREQ,VOLUME)
```

Now that we know how the TI-99/4A works and how to "speak" TI BASIC, let's begin doing some real programming. Use this chapter as a reference. It will be useful as you begin to develop your own programs.





Color Subroutines 3

Now that you have read about how the TI-99/4A works (Chapter 1), and the BASIC commands and statements it uses, it is time to begin working with some subroutines in color. The TI-99/4A is a very colorful computer, and with a little practice, you can dazzle your friends with your color computing ability.

COLOR AND THE TI COMPUTER

Your TI computer can display 16 different colors. Each color has been assigned a number. These numbers are stored permanently in your computer memory. Here are the codes for all 16 colors.

Color	Code No.	Color	Code No.
Transparent	1	Medium Red	9
Black	2	Light Red	10
Medium Green	3	Dark Yellow	11
Light Green	4	Light Yellow	12
Dark Blue	5	Dark Green	13
Light Blue	6	Magenta	14
Dark Red	7	Gray	15
Cyan	8	White	16

If you have ever done any programming, you have already seen two of these colors. While you input a program, the screen is cyan and when the TI runs a program, the screen turns to light green. Now let's see how *we* can control the screen color.

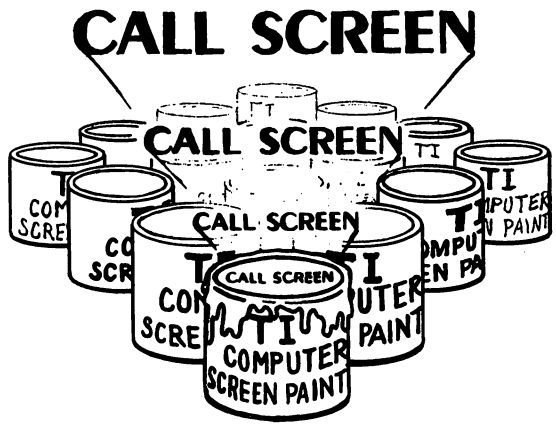
CALL SCREEN

By using one subprogram and the color-code numbers, we can change the screen color to any of the 16 TI colors. Here is how.

The CALL SCREEN subprogram tells the TI-99/4A that we want to change the color of the screen. Now we need to tell the computer what color we want the screen to be. We do this by using a color-code number for the desired color. Input the following:

```
100 CALL CLEAR
110 CALL SCREEN(16)
120 GOTO 110
```

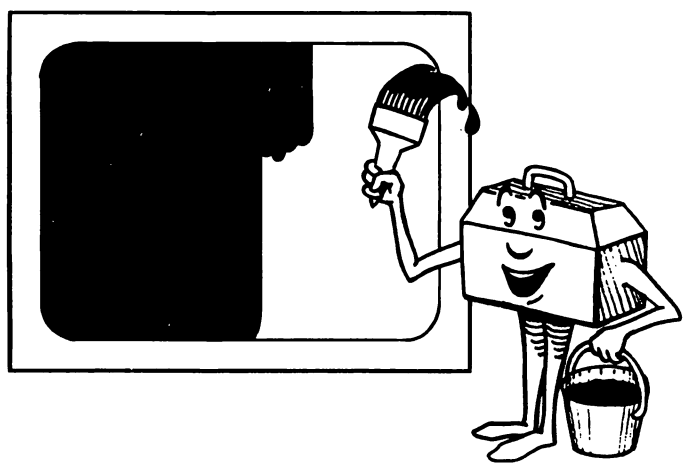
-Note that the code number must be in parentheses.



There are 16 paint buckets; one for each color.

In this program, line 100 clears the screen. Line 110 calls screen color 16 (white). Line 120 makes the computer loop back to line 110 so the screen will stay white until you hit FUNCTION and the number 4 key (CLEAR).

You can change the (16) in line 110 to any number between 1 and 16 and see all the possible colors. Color number 1 is transparent, so the screen will display in light green.



Tool Kit guy painting the screen (after the first program).

You can also use a variable with CALL SCREEN rather than a number. Try this:

```
100 CALL CLEAR
110 C = 2
120 CALL SCREEN(C)
130 GOTO 110
```

When you run this program, the TI-99/4A will display a black screen. Line 110 tells the computer that C = 2 is wanted; this is the color code for black. Now let's use a variable to display all of the TI colors, one after another. Input the following:

```
100 CALL CLEAR          -Clears the screen.
110 FOR C = 1 TO 16     -Tells the computer that C = color
                         1 to color 16.
120 CALL SCREEN(C)      -Displays the screen.
130 NEXT C              -Gets the next color-code number.
```

The first display will be transparent. It will then go to black, then medium green, and so on, until all 16 colors have been displayed. The FOR NEXT commands are very useful and are easy to use once you understand how they work.

When you run the program as it is, you will notice that the screens change colors very quickly. Let's use a *timer,* in the form of another FOR NEXT combination, to slow the display down. Add these lines and see the difference:

```
125 FOR TIME = 1 TO 300
128 NEXT TIME
```

See the difference?

RANDOM COLOR

In the last program, the colors were displayed in order from 1 through 16. With some minor changes, we can get the colors to display in a random order. Input this:

```
100 CALL CLEAR          -Clears the screen.
110 RANDOMIZE           -Tells the TI-99/4A to randomize.
120 C = INT(16*RND)     -Defines C as a random integer
                         between 0 and 16. (We will use
```

	this notation over and over so you should become familiar with it.)
130 IF C = 0 THEN 180	-Since there is no number code 0, we must give C another value in line 180.
140 CALL SCREEN(C)	-Displays the screen colors.
150 FOR I = 1 TO 300	-Lines 150 and 160 are the timer
160 NEXT I	to hold the colors on the screen.
170 GOTO 100	-Tells the TI computer to go back and do it all again.
180 C = 2	-Value of C if C = 0.
190 GOTO 140	-Plots 2 as the screen color and restarts the loop.

Line 120 tells the TI computer to select a random number from 0 to 16. That number then becomes the color code for line 140. If line 120 selects 0, the program branches to 180 where a new value is given to C. Since 0 is not a color-code number, we must tell the TI-99/4A what to do when the number zero comes up. Now let's look at the other color subprogram.

CALL COLOR

The Call Color subprogram allows you to change the characters on the screen to any color you like. The formula for using Call Color is:

CALL COLOR (Character set number, foreground color, background color)

Let's look quickly at each of these components.

Character Set Number

Just as each color has a code number, so does each and every keyboard character. All of the characters are organized into eight sets, with eight characters in each set. There are also eight special character-code sets so that you can define and color your own characters, but we will discuss the special sets later in the book.

Here are the character-set code numbers for the keyboard characters:

Set 1		Set 2	
Code No.	Character	Code No.	Character
32	(space)	40	(
33	!	41	)
34	"	42	*
35	#	43	+
36	$	44	,
37	%	45	-
38	&	46	.
39	`	47	/

Set 3		Set 4	
Code No.	Character	Code No.	Character
48	0	56	8
49	1	57	9
50	2	58	:
51	3	59	;
52	4	60	<
53	5	61	=
54	6	62	>
55	7	63	?

Set 5		Set 6	
Code No.	Character	Code No.	Character
64	@	72	H
65	A	73	I
66	B	74	J
67	C	75	K
68	D	76	L
69	E	77	M
70	F	78	N
71	G	79	O

Set 7		Set 8	
Code No.	Character	Code No.	Character
80	P	88	X
81	Q	89	Y

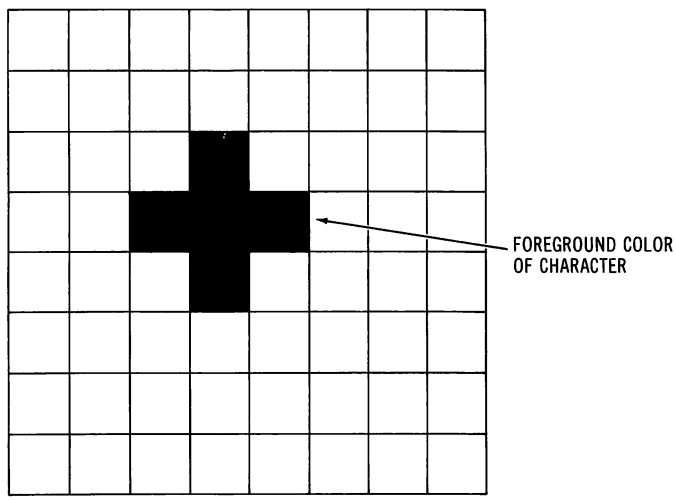
82	R	90	Z
83	S	91	[
84	T	92	\
85	U	93	]
86	V	94	^
87	W	95	—

Now that you know the code numbers, you still have to determine which number is put in the character-set location. If you want to display and color a plus sign (+), look on the character-set chart until you find it. You will see that the ASCII number for a plus sign is 43. Since number 43 is in character set number 2, a number 2 goes into the character-set position in the Call Color subprogram. Like this:

```
CALL COLOR(2,
```

Foreground Color

The second position in the Call Color subprogram defines the foreground color of the character you want to color. The foreground color is the color of the character itself.



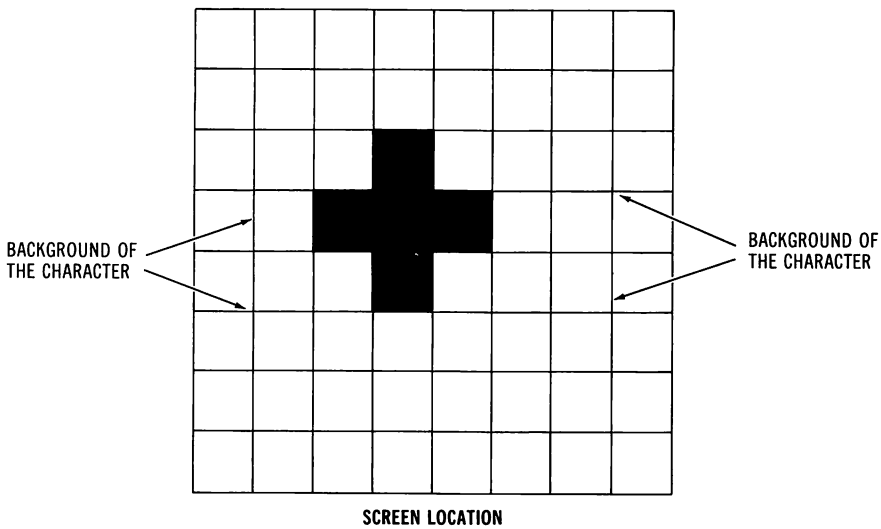
In this case, it is the color of the plus sign (+). Let's say that we want the plus sign to be dark red. The color code for dark red is 7, so a 7 goes

into the foreground color position of our CALL COLOR subprogram; like this:

```
CALL COLOR(2,7
```

Background Color

When a character is displayed, only a portion of the screen location of that character is activated.



If the entire screen location was activated, the character would look like a black square. The background of the character cannot be seen unless we color it. The background color component of the Call Color program lets you change the color of the character's background. Let's say that we want our plus sign to be dark red with a white background. We add 16, or the color code for white, to our Call Color subprogram like this:

```
CALL COLOR(2,7,16)
```

Our line is now complete. This line statement will color the plus sign and any other character in Character Set 2 as red on a white background. Let's look at the Call Color subprogram in action. Input:

```
100 CALL CLEAR          -Clears the screen.
```

```
110 CALL COLOR(2,7,16)
120 PRINT "+++++"
130 GOTO 110
```

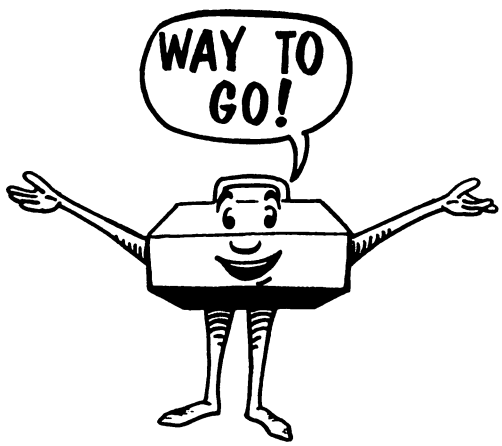
-Colors Character Set 2 as red on a white background.

See how the red plus signs are displayed in little white squares. The white squares are the background of the + sign character. If you want the white squares to go away, change the background color in line 110 to light green, or a 4, like this:

```
CALL COLOR(2,7,4)
```

You can use variables with the Call Color subprogram, just as we did with Call Screen. We can also use Call Screen and Call Color together, as you will see later.

Let's look at a subroutine that you can use as a reward for a correct answer in a game or a quiz. This program uses Call Color, but uses variables instead of numbers.



Reward Subroutine

```
100 CALL CLEAR
110 PRINT "**********
    **********"
120 PRINT "**********
    **********"
130 PRINT "**********
    **********"
```

-Clears the screen.

-Lines 110 to 130 will each print 27 asterisks.

```
140 PRINT "WAY TO GO,
    YOU GOT IT RIGHT"
150 PRINT "**************
    ***********"
160 PRINT "**************
    ***********"
170 PRINT "**************
    ***********"
180 FOR FC = 2 TO 16
190 FOR SET = 1 TO 8
200 CALL COLOR(SET,FC,1)
210 NEXT SET
220 NEXT FC
```

-Prints the message.

-Lines 150 to 170 will each print 27 asterisks.

-Tells the computer that the foreground color = color code numbers 2 to 16 (transparent is left out).

-Tells the computer that the character set = 1 through 8, or all the alpha and numeric characters.

-Calls the asterisks and the message displaying all foreground colors, one at a time, on a transparent background.

-Tells the computer to get the next character set.

-Tells the computer to get the next foreground color.

Line 180 tells the computer the values of all the foreground colors. Line 190 tells the computer to color all the characters in sets 1 through 8.

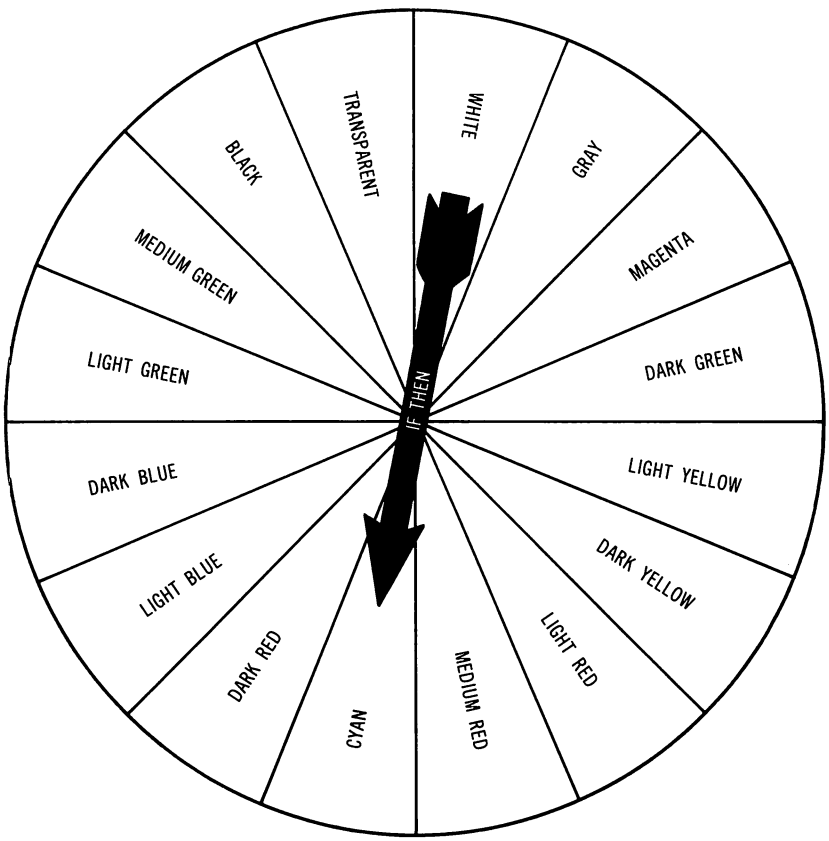
You could also use a variable to change the background color. You could add

```
195 FOR BC = 16 TO 1 STEP -1
205 NEXT BC
```

and the foreground and background colors would change. BC will begin at 16 and go to 1 and FC will begin at 1 and go to 16. You will notice that the colors change much slower with lines 195 and 205 added; that is because we have added another FOR NEXT cycle.

COLOR CHOICE

Now let's put together Call Color, Call Screen, and what we have learned in this chapter to create what we call the Color Choice. The basis of this program is the following subroutine.




```
100 CALL CLEAR
110 CALL SCREEN(16)          -Displays a white screen.
120 FOR SET = 1 TO 8         -Defines set as all characters.
130 CALL COLOR(SET,2,16)     -Calls characters in black on a
140 NEXT SET                  white background.
```

```
150 PRINT "THIS IS A WHITE
    SCREEN WITH BLACK
    LETTERS"
160 FOR I = 1 TO 400          -Lines 160 and 170 are the timer
170 NEXT I                     to hold the message on the
                               screen.
```

Color Choice asks you to type in a color and then the program branches to the inputted color and displays the screen and the message. Here we go. Input this Color Choice program:

```
100  CALL CLEAR
110  PRINT "DO YOU LIKE       -Prints the message.
     BLACK, WHITE, RED,
     CYAN, GREEN, YEL
     LOW, OR BLUE?"
120  A1$ = "BLACK"            -Lines 120 to 180 define A$1 -A$7.
130  A2$ = "WHITE"
140  A3$ = "RED"
150  A4$ = "CYAN"
160  A5$ = "GREEN"
170  A6$ = "YELLOW"
180  A7$ = "BLUE"
190  PRINT "WHAT IS YOUR      -Prints the message.
     FAVORITE COLOR?"
200  INPUT B$                 -Gives the input prompt and
                               allows you to type in a color.
210  IF B$ = A1$ THEN 1000    -If B$ is black go to 1000.
220  IF B$ = A2$ THEN 2000    -If B$ is white go to 2000.
230  IF B$ = A3$ THEN 3000    -If B$ is red go to 3000.
240  IF B$ = A4$ THEN 4000    -If B$ is cyan go to 4000.
250  IF B$ = A5$ THEN 5000    -If B$ is green go to 5000.
260  IF B$ = A6$ THEN 6000    -If B$ is yellow go to 6000.
270  IF B$ = A7$ THEN 7000    -If B$ is blue go to 7000.
1000 CALL CLEAR
1010 CALL SCREEN(2)           -Displays black.
1020 FOR SET = 1 to 8
1030 CALL COLOR(SET,16,2)
1040 NEXT SET
```

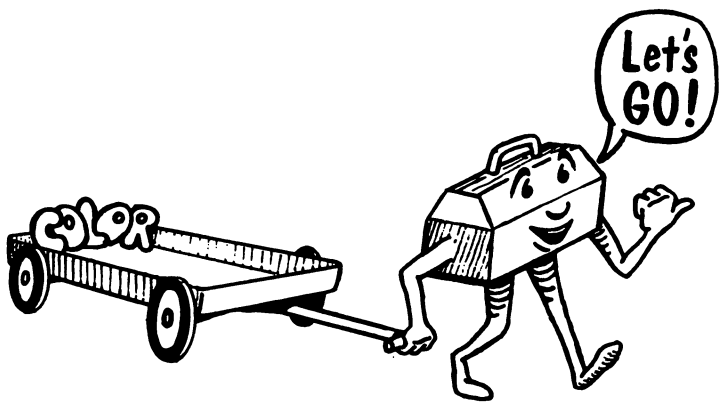
```
1050 PRINT "THIS IS A BLA
     CK SCREEN WITH WH
     ITE LETTERS"
1060 FOR I = 1 TO 400
1070 NEXT I
1080 CALL CLEAR
1090 GOTO 190
2000 CALL CLEAR
2010 CALL SCREEN(16)          -Displays white.
2020 FOR SET = 1 to 8
2030 CALL COLOR(SET,2,16)
2040 NEXT SET
2050 PRINT "THIS IS A WHI
     TE SCREEN WITH BLA
     CK LETTERS"
2060 FOR I = 1 TO 400
2070 NEXT I
2080 CALL CLEAR
2090 GOTO 190
3000 CALL CLEAR
3010 CALL SCREEN(7)           -Displays red.
3020 FOR SET = 1 TO 8
3030 CALL COLOR(SET,2,7)
3040 NEXT SET
3050 PRINT "THIS IS A RED
     SCREEN WITH BLACK
     LETTERS"
3060 FOR I = 1 to 400
3070 NEXT I
3080 CALL CLEAR
3090 GOTO 190
4000 CALL CLEAR
4010 CALL SCREEN (8)          -Displays cyan.
4020 FOR SET = 1 TO 8
4030 CALL COLOR(SET,7,8)
4040 NEXT SET
4050 PRINT "THIS IS A CY
     AN SCREEN WITH RED
     LETTERS"
4060 FOR I = 1 TO 400
```

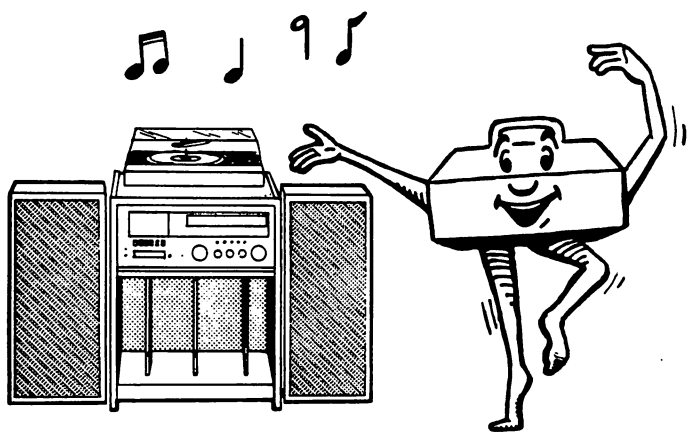
```
4070 NEXT I
4080 CALL CLEAR
4090 GOTO 190
5000 CALL CLEAR
5010 CALL SCREEN(13)          -Displays green.
5020 FOR SET = 1 TO 8
5030 CALL COLOR(SET,7,13)
5040 NEXT SET
5050 PRINT "THIS IS A GRE
     EN SCREEN WITH CY
     AN LETTERS"
5060 FOR I = 1 TO 400
5070 NEXT I
5080 CALL CLEAR
5090 GOTO 190
6000 CALL CLEAR
6010 CALL SCREEN(1)           -Displays yellow.
6020 FOR SET = 1 TO 8
6030 CALL COLOR(SET,13,11)
6040 NEXT SET
6050 PRINT "THIS IS A YEL
     LOW SCREEN WITH
     GREEN LETTERS"
6060 FOR I = 1 TO 400
6070 NEXT I
6080 CALL CLEAR
6090 GOTO 190
7000 CALL CLEAR
7010 CALL SCREEN(5)           -Displays blue.
7020 FOR SET = 1TO 8
7030 CALL COLOR(SET,11,5)
7040 NEXT SET
7050 PRINT "THIS IS A BLUE
     SCREEN WITH YEL
     LOW LETTERS"
7060 FOR I = 1 TO 400
7070 NEXT I
7080 CALL CLEAR
7090 GOTO 190
```

We have used IF/THEN statements to branch the program. The IF/THEN statements allow you to branch to another part of the program if a certain condition exists.

You can change the colors to include all 16 colors if you like, or you might want to add some fancy screen prompts at the beginning of the program.

We have discussed some simple color subroutines and shown you how to build a program using one of them. We recommend that you experiment with putting pieces of programs together to build your own program. Try one using the random-screen color subroutines. When you are ready, we will move on to Chapter 4 and talk about sound.





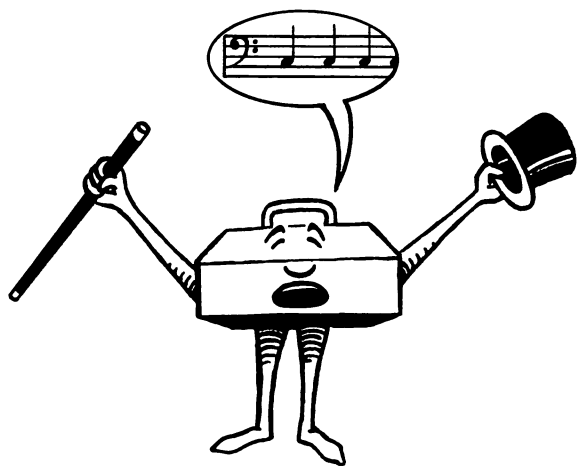
Sound Subroutines 4

Your TI-99/4A computer has a great voice. It has a musical range greater than any two musical instruments we can name. You can learn to "play" the TI computer with a minimum of practice. In this chapter, we will show you how to make beautiful music with your TI-99/4A and how to use sound subroutines to add some pizzazz to your future programs. Let's get started.

CALL SOUND

The CALL SOUND subprogram will tell your TI computer that you want it to play either a tone or one of eight noises. CALL SOUND is not a standard BASIC language command. The words, CALL SOUND, trigger a preprogrammed set of instructions that the TI-99/4A understands. These instructions are manufactured into the design of the machine. That's why we call CALL SOUND a subprogram and not a command. Let's see how it works. Input the following line

```
CALL SOUND(1000,262,1)ENTER
```



You have just played a 1-second rendition of Middle C on the musical scale at Volume one. The three numeric values in parentheses tells the computer how long to play the note (Duration), which note to play (Frequency), and how loud to play the note (Volume). To help you use this subprogram, let's work with each value for a moment.

If you want to think of the values as a formula or a recipe it may help you understand.

CALL SOUND(DURATION,FREQUENCY,VOLUME)

Duration

The first value in the parentheses tells the computer how long to hold the selected note. The TI-99/4A computer can hold a tone for as little as 0.001 second or as long as 4.25 seconds. Since the computer works and reacts quicker than we humans, the computer measures time in smaller segments. For this subprogram, we have to think in terms of milliseconds. There are 1000 milliseconds in one second. The *duration value* must be stated in milliseconds. So, if you want a tone to play for 2 seconds, you should enter 2000 in the duration location. Let's look at this in table form:

Desired Playing Time	Time in Milliseconds	Proper Duration Entry
½ second	500	500
1 second	1,000	1000
2 seconds	2,000	2000
3 seconds	3,000	3000
4 seconds	4,000	4000

Enter the following:

```
CALL SOUND(100,262,1)ENTER
CALL SOUND(3000,262,1)ENTER
CALL SOUND(4250,262,1)ENTER
```

Hear the difference? Now let's look at the next value in the CALL SOUND subprogram.

Frequency

The tones that you are creating are electronic variations. These variations or frequencies are measured in hertz. Hertz is a term used by technical people in the electronics field. The word hertz is abbreviated to Hz. The frequency range for the TI-99/4A goes from a very low pitch of 110 Hz to an extremely high pitch of 44,733 Hz.

Input the following line:

```
CALL SOUND(1000,44733,1)ENTER
```

Unless you have superhuman hearing you should not be able to hear this tone at all. In fact, the highest frequency we can hear is around 14,900 Hz. Now would be a good time to listen to some TI computer *tones*. Input:

```
CALL SOUND(1000,110,1)ENTER      -110 = the low range.
CALL SOUND(1000,4000,1)ENTER
CALL SOUND(1000,392,1)ENTER      -392, = G above Middle C.
```

The TI-99/4A can make lovely music, but it can also make eight noises that are not at all musical. To make these noises, use the numbers −1 through −8 as the tone frequencies. These noises are terrific for use in the games we will develop later.

Let's try a noise. Input:

```
CALL SOUND(1000,−4,1)ENTER
```

We think this sounds a lot like Pac Man eating his way around the screen.

You should try the other noises if you haven't already.

Volume

The last value in the CALL SOUND subprogram is Volume. You guessed it. This value tells the computer how loud or soft to play the tone or noise. The volume range for the TI-99/4A runs from an input value of 0 (loudest) to 30 (softest). We advise that you set the volume on your tv monitor to a low setting when playing music at level 0 or 1.

Now, let's listen to the whisper level of the TI computer. Input:

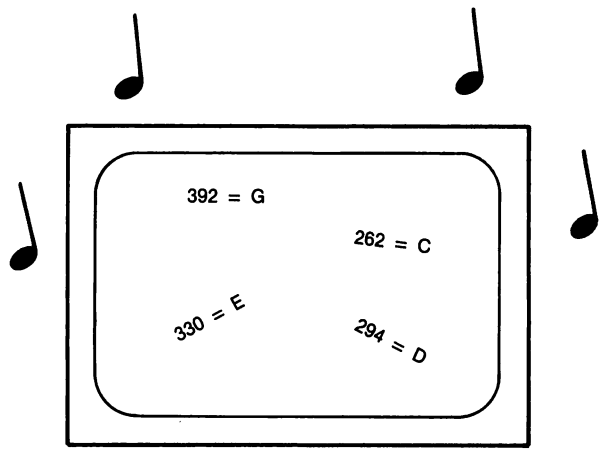
```
CALL SOUND(1000,262,30)ENTER
```

If you can't hear this note, turn up your tv volume. This is really a whisper.

Until now, we have been using the CALL SOUND subprogram in the immediate mode, which is fine for listening to one tone at a time. When CALL SOUND is used as a program statement, a lot of programming excitement begins.

CALL SOUND IN A PROGRAM

To begin, let's look at a little program that allows you to call up any sound by just inputting two values. We call this program the Sound Encyclopedia. Input the following lines:




```
5  REM SOUND ENCYCLO
   PEDIA
10 CALL CLEAR
20 INPUT "NOTE": N

30 INPUT "VOLUME": V

40 CALL SOUND(500,N,V)


50 GOTO 20


RUN
```

-Names the program.

-Clears the screen.

-Gives a screen prompt and allows you to enter a frequency.

-Gives a prompt and allows you to enter a volume.

-Tells the computer to play the frequency and volume input in lines 20 and 30 for ½ second.

-Loops back to line 20 so that you can input more frequencies and volume levels.

NOTE: Since this is a continuous loop program you will have to hit the Clear key (the FUNCTION and 4 keys) to stop the program.

This little program points out a neat feature of CALL SOUND. You can define each value as a variable just as we did in lines 20 and 30. This makes programming much easier. We have still stated all three values for CALL SOUND, but we used the letters N and V instead of numbers.

CALLING MORE THAN ONE SOUND

The CALL SOUND subprogram allows us to call more than one sound at a time. In fact, you can call up to three tones and one noise at a time. Here's how we do it. Input:

```
CALL SOUND (1000,262,1,330,1,392,1)
```

All the tones will be of the same duration, and the computer will play 262,330,392 at a volume setting of "1". Once we set the duration, we can enter frequency and volume settings for three tones. The volume values can be varied.

Here is the formula for entering more than one sound. For 2 sounds, input

```
CALL SOUND(Dur,Freq,Vol,Freq,Vol)
```

For 3 sounds, input

```
CALL SOUND(Dur,Freq,Vol,Freq,Vol,Freq,Vol)
```

Finally, for 3 sounds plus 1 noise, input

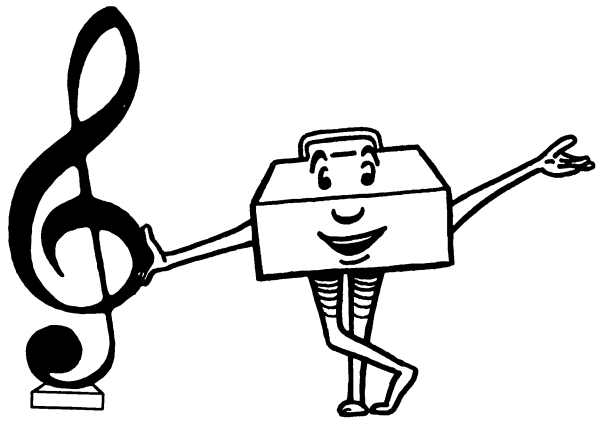
```
CALL SOUND(Dur,Freq,Vol,Freq,Vol,Freq,Vol,-Freq,Vol)
```

This terrific feature allows you to quickly program notes and musical chords so that you can play real music with the TI computer. Let's play some music.

MUSIC SUBROUTINES

Let's begin our music lesson with a scale. This scale is not exactly musically correct, but it is an easy way of approximating a scale if you want a scale-like sound effect for a game or quiz.

Input the following program:



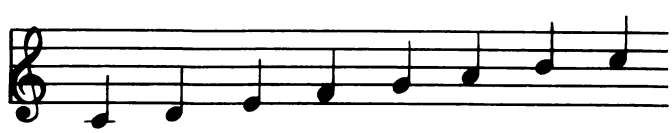
```
10 TONE = 226                    -Defines the first frequency for
                                  line 30.
20 FOR SCALE = 1 TO 8            -Tells the computer that it will
                                  play 8 tones.
30 CALL SOUND(2500,TON           -Tells the computer to play the
   E,1)                           tone for 2½ seconds at level 1.
40 TONE = TONE + 18              -Beginning at 226, the computer
                                  will play every eighteenth fre-
                                  quency.
50 NEXT SCALE                    -Moves to next note.
RUN
```

The TI-99/4A will automatically play 8 notes, 18 frequencies apart, for a duration of 2½ seconds, at a volume level of 1.

As we will soon see, a loop will help us later on. Let's put a continuous loop in this program. Input this:

```
60 GOTO 10
```

This will make the approximate scale play forever unless you hit CLEAR. Let's try a real scale now.



C SCALE

THE C SCALE

Type the following program.

```
5  CALL CLEAR
10 T = 1000
20 V = 1
30 C = 262
40 D = 294
50 E = 330
60 F = 349
70 G = 392
80 A = 440
90 B = 494
100 CALL SOUND (T,C,V)
110 CALL SOUND (T,D,V)
120 CALL SOUND (T,E,V)
130 CALL SOUND (T,F,V)
140 CALL SOUND (T,G,V)
150 CALL SOUND (T,A,V)
160 CALL SOUND (T,B,V)
```

-States the duration variable for future CALL SOUND lines 100-160.

-States the volume level variable for lines 100–160.

-Lines 30 through 90 state the frequency variables for lines 100-600.

-Lines 100 through 160 call and play each tone in the C scale.

This little program will play the C scale one tone at a time. After all seven notes have been played, the program will stop. How would you get the scale to play again and again? If you said "Add a GOTO statement at the end of the program," you guessed right. Add:

```
170 GOTO 100
```

and the scale will play over and over again.

You can tell the computer to play any musical scale using this subroutine as an example. All you need to do is change the order of the notes played in lines 100 through 160 and, in some cases, the frequencies of the notes. To play the G scale, you must change the order of the CALL SOUND statements to:

```
100 CALL SOUND(T,G,V)
110 CALL SOUND(T,A,V)
```

```
120 CALL SOUND(T,B,V)
130 CALL SOUND(T,C,V)
140 CALL SOUND(T,D,V)
150 CALL SOUND(T,E,V)
160 CALL SOUND(T,F,V)        -This note will actually be F sharp.
```

You will also need to change the frequencies of all the notes above G in the C scale (i.e., C, D, E, and F).

Here are the new frequency variables:

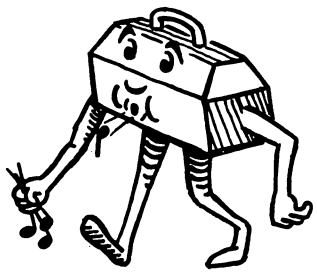
```
C = 523
D = 587
E = 659
F = 740
G = 392
A = 440
B = 494
```

These variables and the new CALL SOUND order will give you the G Major scale. The value of F is actually F Sharp, which is the seventh note on the scale. You can put the two scales together if you like. Just define all of the necessary frequencies and be sure you note the difference in high and low notes. For example, call Middle C "C" and High C "H.C." Try combining these two scales and experiment with a variety of note combinations. After you are finished experimenting, read on and we will show you how to add harmony to your TI-99/4A computer.

PLAYING CHORDS ON THE TI-99/4A

As we told you before, you can generate more than one tone in a single CALL SOUND statement. Remember you must state the Duration first, then the Frequency, and then the Volume for each tone that you want—up to a total of three. Here is the formula:

```
D = DURATION
F = FREQUENCY
V = VOLUME
                CALL SOUND(D,F,V,F,V,F,V)
```



THE TI COMPUTER PLAYS CHORDS

In the next subroutine, we are going to make the TI computer sound like an organ. Enter the following program.

```
5   CALL CLEAR
10  S = 1500          -States Duration variable—1.5
                       second.
20  V = 1             -States Volume level at 1.
30  C = 262           -Lines 30 through 90 state the
40  D = 294            frequency values of notes C
50  E = 330            through B.
60  F = 349
70  G = 392
80  A = 440
90  B = 494
100 F1 = 370          -States the value of F Sharp below
                       Middle C.
110 G1 = 208          -States the value of G Sharp
                       below Middle C.
120 A1 = 466          -Value of A Sharp above Middle
                       C.
130 HD = 587          -Value of D above High C.
140 C1 = 554          -Value of High C Sharp.
150 HE = 659          -Value of E above High C.
160 D1 = 622          -Value of D Sharp above High C.
170 F2 = 740          -Value of F Sharp above High C.
200 CALL SOUND(S,C,V,E,V,  -Plays C Chord.
    G,V)
300 CALL SOUND(S,D,V,F1,   -Plays D Chord.
    V,A,V)
```

```
400 CALL SOUND(S,E,V,G1,      -Plays E Chord.
    V,B,V)
500 CALL SOUND(S,F,V,A1,      -Plays F Chord.
    V,HD,V)
600 CALL SOUND(S,G,V,B,V,     -Plays G Chord.
    D,V)
700 CALL SOUND(S,A,V,C1,      -Plays A Chord.
    V,HE,V)
800 CALL SOUND(S,B,V,D1,      -Plays B Chord.
    V,F2,V)
RUN
```

This program will play a lovely selection of chords. Save this program for later use, because we are going to use it several times in this chapter.

How can we add to this program to make it even more interesting? What if we play the notes or chords in a set order instead of letting the computer have all the fun? Then, we can play the TI-99/4A just like a musical instrument.

THE TI ORGAN

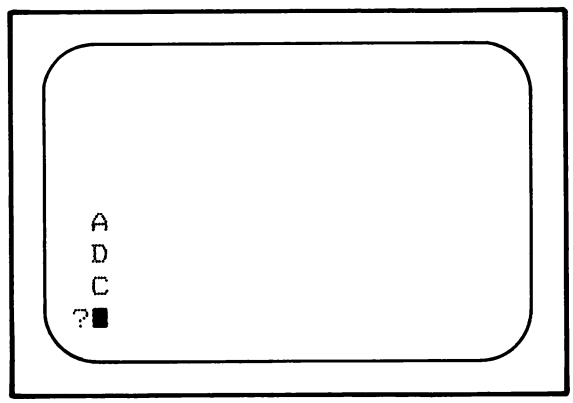
Using our chord program, let's make some useful changes. We are going to tell the computer that we will input the chord that we want to play. This is one way to do it.

```
171 INPUT A$                  -This tells the computer to give
                               us a prompt and allow us to
                               make a key selection.
172 IF A$ = "C" THEN 200      -If we input C on the keyboard,
                               the computer will go to line 200
                               and play the tones listed there.
173 IF A$ = "D" THEN 300      -If we input D, the computer will
                               play the tones in line 300.
174 IF A$ = "E" THEN 400      -Get the idea?
175 IF A$ = "F" THEN 500
176 IF A$ = "G" THEN 600
177 IF A$ = "A" THEN 700
178 IF A$ = "B" THEN 800
```

With these commands, we have told the computer that we will input a note but, as the program is now, the TI computer will play the note we select and, then, every note thereafter until the program ends. We need to get the computer to play just one note and allow us to then select another. We can easily get the job done with a few well-placed GOTO statements. Input the following statements:

```
230 GOTO 171
330 GOTO 171
430 GOTO 171
530 GOTO 171
630 GOTO 171
730 GOTO 171
830 GOTO 171
```

With these statements in place, the computer will play the chord we tell it to play and will then go back to allow us to pick another chord.



A FEW HINTS ABOUT PLAYING THE TI ORGAN

You must always press ENTER after the letter key. Since the chords play for a second and a half, you can play without interruption with just a little practice. As soon as you hear the chord, input your next desired chord. The TI-99/4A thinks pretty fast and it will accept the input as soon as the preceding chord begins to play.

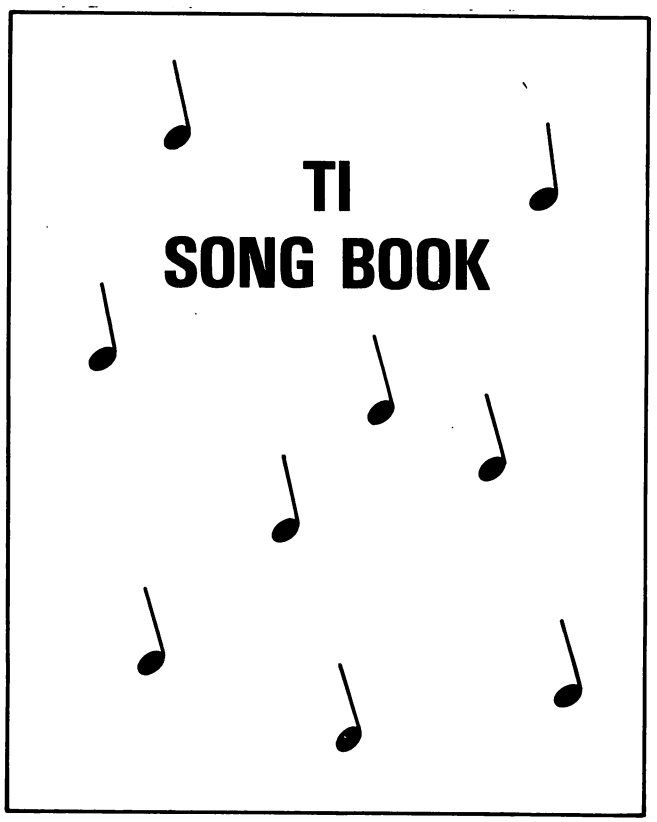
This program can be altered to play only one note at a time like a piano. If you want to play the piano, simply delete the note values that

you don't want, and change the CALL SOUND lines so that only one note per line is stated.

Later in this chapter, we will show you how to use the TI computer's musical ability, in combination with the color subroutines that we worked with in Chapter 3. But right now, let's program the TI-99/4A to play some simple songs all by itself.

THE TI SONG BOOK

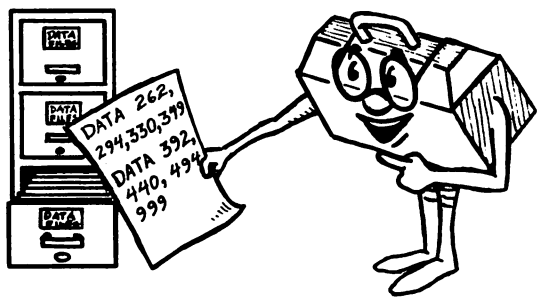
By playing one note at a time in the proper order, you can get your TI computer to play any song that you are willing to program. If we were to use the normal CALL SOUND subprograms that we have used up to now, we would need to enter a CALL SOUND statement for every note in a song. Even a short song would require a very long and a very dull program. The TI-99/4A computer has a better way.



SOUND SUBROUTINES USING DATA FILES

The DATA statement and the READ statement can often be used to save a lot of inputting time. Values can be assigned to variables by telling the computer to read from a DATA list. However, the DATA statement does not tell the computer to do anything but read the values in the list.

When we use the READ command and a DATA file, we must keep some things in mind. The computer begins with the first value in a data list and continues through until there is no more data or until the computer finds a value that tells it to stop. So, be sure you input the correct data and be sure that the notes are in the right order, because if you don't, the TI computer will play some sour notes.



Let's use the READ command and DATA files to make some beautiful music with the TI-99/4A computer. Input the following:

```
10 CALL CLEAR
20 READ F
30 IF F = 999 THEN 60
40 CALL SOUND (500,F,1)
50 GOTO 20
```

-This is the READ command. It tells the computer to read the files for the values of F.

-999 is the last item in the file. If you don't use a terminal item, the TI computer will print an error statement after it plays the last note.

-Calls the notes.

-Forms a loop so the computer can read the next value.

```
60 PRINT "I AM FINISHED
   NOW"
70 GOTO 70

80 DATA 262,262,392,392,
   440
90 DATA 440,392,349,349,
   330
100 DATA 330,294,294,262,
   392
110 DATA 392,349,349,330,
   330
120 DATA 294,392,392,349,
   349
130 DATA 330,330,294,262,
   262
140 DATA 392,392,440,440,
   392
150 DATA 349,349,330,330,
   294
160 DATA 294,262,999
```

-When item 999 is read, this line will print.

-Holds "I AM FINISHED NOW" on the screen by creating an endless loop.

-Lines 80 through 160 are the frequencies, listed in order, to play the song. The computer just keeps reading until it comes to the last file 999.

When you run this beauty, you will hear an electronic rendition of Twinkle, Twinkle, Little Star. Save this program because we will show you some graphic subroutines to use with this song—later in Chapter 6.

Here are the note sequences of some other favorites that you can try.

Mary Had a Little Lamb

E D C D E E E D D D E G G E D C D E E E E D D E D C

Are You Sleeping

G A B G G A B G B C D B C D D E D C B G D E D C B G G D G G D G

We will use the DATA and READ statements more in later chapters. But, for now, remember that these commands are great time savers.

ARCADE SOUNDS

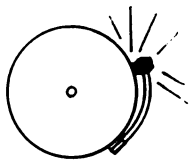
Earlier in this chapter, we mentioned the noise capability of the TI computer. All of the programs in this chapter are useful and can be applied to other programs, but the arcade noises will be particularly useful. Type in the following:

```
5  CALL CLEAR
10 CALL SOUND (500,-2,2)
20 CALL SOUND (500,-1,2)
30 GOTO 10
```

Sounds like a police siren doesn't it? This sound can be used effectively in a variety of chase games.

Need a burglar alarm or warning tone for a game? Try this:

```
5  CALL CLEAR
10 CALL SOUND (100,-3,2)
20 CALL SOUND (100,-2,1)
30 CALL SOUND (100,-1,1)
40 GOTO 10
```



You can change the duration and volume values to change the effect.

If you are going to develop any type of space game, you must have a rocket blast-off effect. Here is a nice rocket blast subroutine.

```
5  CALL CLEAR
10 FOR WARN = -3 TO -1
   STEP 1
```

-Lines 10 and 20 sound the warning buzzer.

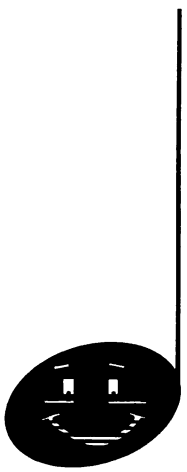
```
20 CALL SOUND (1000,
   WARN,2)
30 NEXT WARN
40 FOR BLAST = -7 TO -5
   STEP 1
50 CALL SOUND(1000,
   BLAST,0)
60 NEXT BLAST
70 CALL SOUND(2000,-1,1)
```

-Lines 40 to 70 cause the Blast Off.

If you want to make a PAC MAN munching noise, all you have to do is add this line to your program:

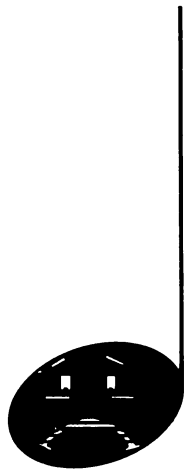
```
10 CALL SOUND(1000,-4,2)
```

How about some musical reinforcement for a correct answer in a quiz.



Try this:

```
5   CALL CLEAR
10  T = 250
20  V = 1
30  C = 262
40  D = 294
50  B = 247
60  E = 330
70  CALL SOUND(T,C,V)
80  CALL SOUND(T,D,V)
90  CALL SOUND(T,E,V)
100 CALL SOUND(T,D,V)
110 CALL SOUND(T,E,V)
120 CALL SOUND(T,C,V)
```



And if you enter an answer incorrectly, your TI computer will tell you with a sound like the following:

```
10 T = 250
20 V = 1
30 C = 262
40 D = 294
50 B = 247
60 D1 = 311
70 CALL SOUND(T,C,V)
```

```
80  CALL SOUND(T,C,V)
90  CALL SOUND(T,C,V)
100 CALL SOUND(T,D1,V)
110 CALL SOUND(T,D,V)
120 CALL SOUND(T,D,V)
130 CALL SOUND(T,C,V)
140 CALL SOUND(T,C,V)
150 CALL SOUND(T,B,V)
160 CALL SOUND(T,C,V)
```

The more you work with the TI computer's "voice," the more interesting touches you can add to your programs. Keep experimenting.

PUTTING COLOR AND SOUND TOGETHER

We covered color in Chapter 3 and, now, we have covered sound. This is a good time to combine what we have learned into one program.

Let's take the chord organ program and add some color to it.

```
10  S = 1500                    -Value of chord duration.
20  V = 1                       -Volume.
30  C = 262                     -Lines 30 through 170 are the
40  D = 294                      frequency values for all the
50  E = 330                      notes.
60  F = 349
70  G = 392
80  A = 440
90  B = 494
100 F1 = 370
110 G1 = 208
120 A1 = 466
130 HD = 587
140 C1 = 554
150 HE = 659
160 D1 = 622
170 F2 = 740
171 INPUT A$                    -Allows us to input a chord.
172 IF A$ = "C" THEN 200        -Lines 172 through 175 branch to
173 IF A$ = "D" THEN 300         the correct line depending on
174 IF A$ = "E" THEN 400         which chord we select.
```

```
175 IF A$ = "F" THEN 500
176 IF A$ = "G" THEN 600
177 IF A$ = "A" THEN 700
178 IF A$ = "B" THEN 800
190 CALL CLEAR
200 CALL SOUND(S,C,V,E,V,
    G,V)
210 CALL SCREEN(15)
220 CALL CLEAR
230 GOTO 171
300 CALL SOUND(S,D,V,F1,
    V,A,V)
310 CALL SCREEN(2)
320 CALL CLEAR
330 GOTO 171
400 CALL SOUND(S,E,V,G1,
    V,B,V)
410 CALL SCREEN(3)
420 CALL CLEAR
430 GOTO 171
500 CALL SOUND(S,F,V,A1,
    V,HD,V)
510 CALL SCREEN(5)
520 CALL CLEAR
530 GOTO 171
600 CALL SOUND(S,G,V,B,V,
    D,V)
610 CALL SCREEN(7)
620 CALL CLEAR
630 GOTO 171
700 CALL SOUND(S,A,V,C1,
    V,HE,V)
710 CALL SCREEN(11)
720 CALL CLEAR
730 GOTO 171
800 CALL SOUND(S,B,V,D1,
    V,F2,V)
810 CALL SCREEN(14)
820 CALL CLEAR
830 GOTO 171
```

-Calls the C chord.

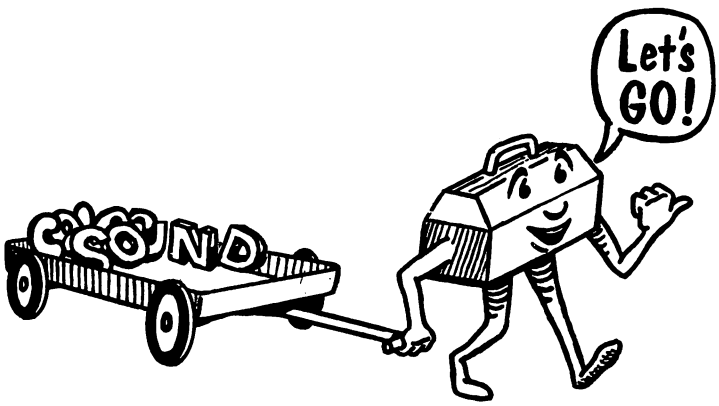
-Colors the screen white.

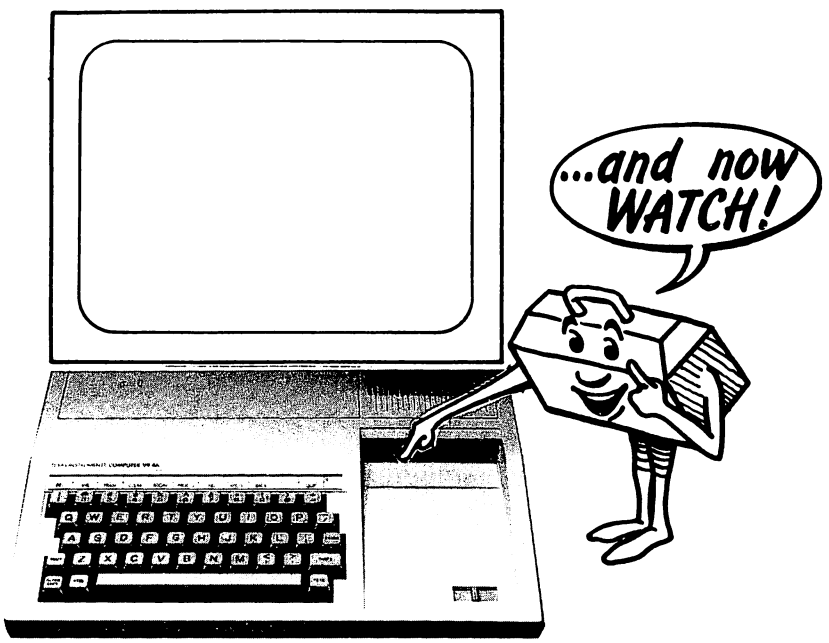
-Loops back.

-The rest of the program is a repeat of lines 200–230. Each group plays a different chord with a different color.

With the addition of some CALL SCREEN commands, we have added a splash of color to our electronic organ. Each time a chord is played, the screen will flash in the color shown in lines 210, 310, 410, 510, 610, 710, and 810. You can add more color and more effects using the Call Color subprogram discussed earlier.

In this chapter we have talked about a lot of ways to use sound in your TI programs. We encourage you to experiment and find even more sound subroutines to use.





Graphics Subroutines 5

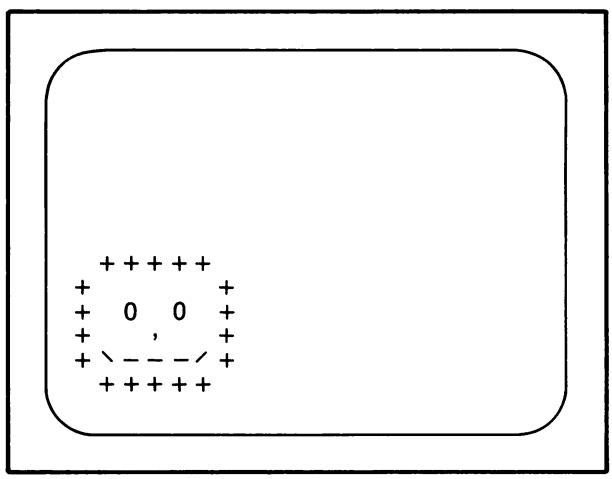
One of the main reasons that you probably bought your TI-99/4A was so that you could create computer graphics. Right? Well, in this chapter, we will show you how to create borders, graphs, playing boards for games, and a variety of characters and techniques that are both fun and very useful.

CREATING GRAPHICS WITH THE PRINT COMMAND

Graphics can be created using the PRINT command. You can draw a design using the characters available on the keyboard. Try this.

```
5  CALL CLEAR
10 PRINT " +++++ "          -space, 5 pluses, space.
20 PRINT "+       +"        -1 plus, 5 spaces, 1 plus.
30 PRINT "+ O O +"          -1 plus, space, O, space, O, space,
                              1 plus.
40 PRINT "+   ,   +"        -1 plus, 2 spaces, comma, 2
                              spaces, 1 plus.
50 PRINT "+\- - -/+"        -1 plus, backwards slash, 3 dashes,
                              slash, 1 plus.
60 PRINT " +++++ "          -space, 5 pluses, space.
70 PRINT GOTO 10
```

This program will display a smiling face at the bottom left side of the



screen when you type RUN and ENTER. Line 70 makes the face print over and over and scroll off the top of the screen. You can make almost anything print on the screen using the PRINT command, but it will always print in the same place. You can use PRINT TAB to move the graphic over to the right, but it will always display at the bottom. The TI-99/4A also has many other graphics capabilities.

But, before we can create other graphics on the TI computer, we need to understand how the computer "sees" the tv screen. Just as we must speak to the computer in a language that it understands (TI BASIC), we must learn to see the screen locations as the computer "sees" them.

SCREEN LOCATIONS

When you turn the TI-99/4A on you see a light blue screen with a prompt at the bottom, but the computer "sees" the screen in a much different way. The TI computer "sees" the screen as 768 blocks or squares. These blocks are identified by row numbers and column numbers. The screen display is made up of 24 rows and 32 columns.

Rows

The row designation describes the vertical or up and down location of the blocks. Row 1 is at the top of the screen and row 24 is at the bottom of the screen (Fig. 5-1). When the graphic subprograms are used (we will discuss them later), the row number is the first number listed in the parentheses. Thus, the row number gives the TI-99/4A the vertical screen location.

Columns

The column designation describes the horizontal or side to side location of the blocks. Column 1 is at the far left of the screen and column 32 is at the far right of the screen. In the graphic subprograms that we will discuss later in this chapter, the column number is the second number given in the parentheses. The column number gives the TI computer the horizontal screen location. Using the row and column numbers, you can plot or identify any screen block location on the screen. Find the screen location for the block at row 1, column 1. If you guessed the first block in the upper left-hand corner of the screen, you guessed right. We suggest that you get some graph paper and number the rows and columns using our illustration in Fig. 5-1 as an example.

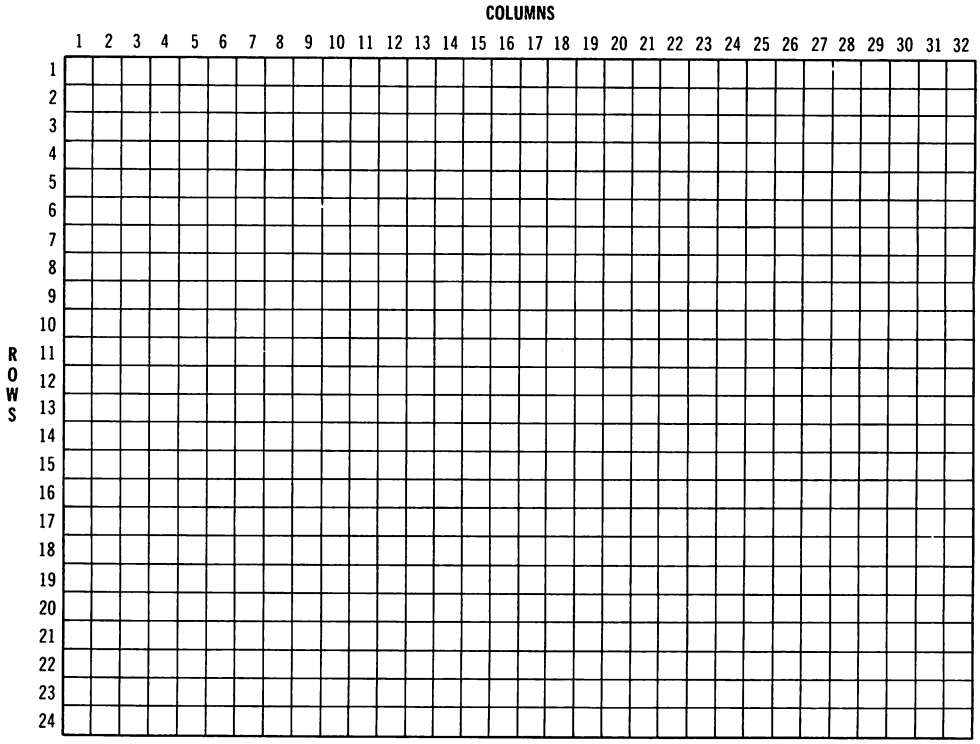


Fig. 5-1. TI-99/4A screen location grid.

This will give you a chance to work out your graphics before you program them.

ASCII CODE

Each character on your TI keyboard has been assigned an ASCII number. This is a predetermined number that the TI computer equates with the keyboard character that you input. For example, the uppercase letter D has the ASCII number 68. These numbers must be used in the graphic subprograms so that the computer knows which character you want to display.

All of the characters on the keyboard use only a portion of one of the 768 blocks on the TI screen. If all of the block is activated, the entire block will be black. So how do we see letters and characters instead of

black blocks? Well, each one of the 768 blocks is made up of 64 smaller blocks—8 across and 8 down. By instructing the computer to darken these smaller blocks, letters and characters are created. For example, the TI computer "sees" the letter A like what is shown in Fig. 5-2.

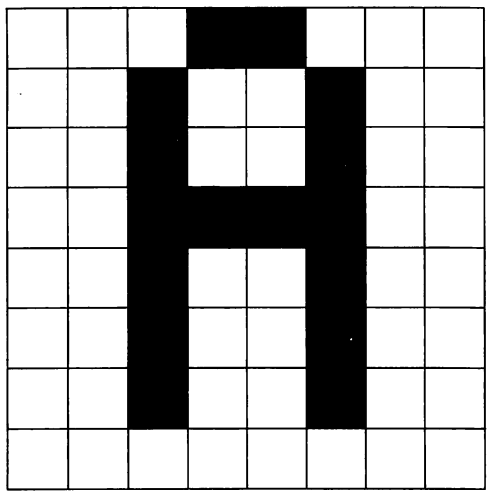


Fig. 5-2. Character definition of the letter "A".

Characters are described to the computer by using a hexadecimal code. The use of the ASCII numbers and how to create your own characters can best be learned by doing. We will tell you how to create your own characters later in the chapter.

CALL CLEAR

The easiest graphics subprogram to use, and perhaps the most useful as well, is CALL CLEAR. CALL CLEAR tells the computer to clear the screen. You can use this subprogram anytime you want the screen to clear. CALL CLEAR will not disturb the program, it only *clears the screen* of any graphic characters. We will use this subprogram a lot in this chapter.

HCHAR

HCHAR is a subprogram that works very much the same way as does CALL COLOR and CALL SOUND. When used with CALL, the HCHAR subprogram will print a character at any screen location that you wish.

The HCHAR subprogram must be followed by a series of numbers in parentheses. Let's look at each number in the command. Input:

```
10 CALL HCHAR(1,5,36)
20 GOTO 10
RUN
```

-Plots a dollar sign at row 1, column 5.

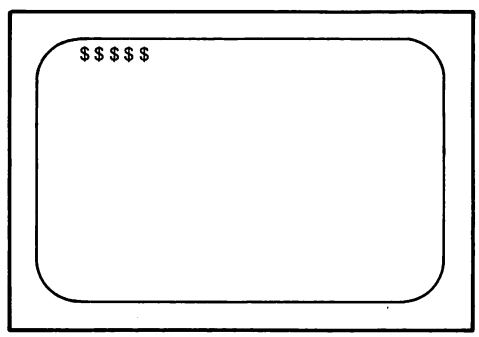
-Creates a loop so the character will remain on the screen.

You should see a dollar sign ($) at screen location row 1, column 5. The first number in the parentheses tells the computer to find row 1 and wait for more directions. The row number is always the first number.

The second number tells the computer to find column 5. The second number must always be the column number. With the row and column number in place, the computer needs to know what should be displayed at the defined location. The third number is the ASCII code for the desired character. In this case, the code is for the dollar sign ($). We have included a complete listing of the ASCII codes in the appendix.

We can add another number after the ASCII code. This fourth number tells the computer to repeat the character. Try this:

```
10 CALL HCHAR (1,5,36,5)
20 GOTO 10
RUN
```



This little program will print a dollar sign at row 1, column 5 and at row 1, columns 6, 7, 8, and 9. This feature allows you to repeat characters as many times as you want. But, remember that the HCHAR repeat function only repeats horizontally, or from side to side, and only from left to right.

Now, input:

```
10 CALL HCHAR(1,5,36,400)
```

The dollar sign just keeps on going, doesn't it? The character will repeat until a row is full and it will then go to the next row.

Try displaying different characters, a row at a time. Begin at row 1, column 1. If you want to fill a 32-column row, you will need to repeat the character 32 times. Here is the beginning of a screen-filling character-display program.

```
10 CALL HCHAR(1,1,34,32)
20 CALL HCHAR(2,1,35,32)
30 CALL HCHAR(3,1,38,32)
```

You can go on if you like.

Remember the following things when using HCHAR(ROW,COL, ASCII,REPEAT).

1. The characters print from the row/column location and from left to right.
2. If you use the repeat option, the character will repeat as many times as you like. When the characters come to the end of a row, they will automatically continue to the next row and will go until they complete the specified number of repeats.

VCHAR

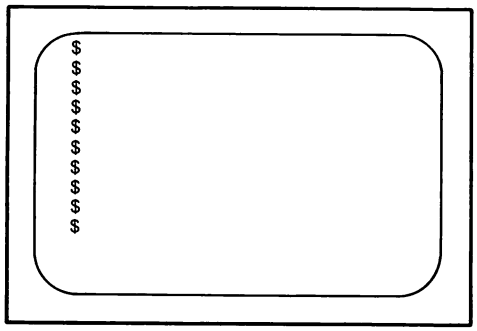
VCHAR is another graphic subprogram that works pretty much the same way that HCHAR does. When used with the CALL command, you can create graphics just as we did with HCHAR. Try this:

```
10 CALL VCHAR(1,5,36)
20 GOTO 10
```

This little program gives exactly the same results as the HCHAR program. The "1" indicates row 1 and the "5" still indicates column 5, and, just as before, the "36" is the ASCII code for the dollar sign. The difference between HCHAR and VCHAR only shows up when the fourth number appears.

Input the following routine:

```
10 CALL VCHAR(1,5,36,10)
20 GOTO 10
```



See? The dollar sign now repeats 10 times vertically, or up and down. VCHAR is used for plotting vertical sequences. Remember that the character will begin at the given row/column coordinate and will plot in a downward direction only.

Now that we know how to position graphics where we want them on the screen, let's look at a way to create our own graphics characters.

CHAR ROUTINES

The CHAR subprogram gives you the chance to create your own graphics characters. To use CHAR, you must first understand a little about hexadecimal coding and the makeup of a single character position.

Character or Screen Positions

Each character position, or row/column coordinate, is made up of 64 tiny boxes—8 boxes across by 8 boxes down. By energizing these tiny boxes, we can create graphics characters not present on the TI keyboard. The trick is to energize the correct boxes in the 64-box grid to get the character you want. This takes a little practice, but it isn't hard to do once you understand how the computer "sees" each character position. The computer "sees" the 64-box grid as if it were divided into two blocks of 32 boxes (4 boxes across and 8 rows down), like that shown in Fig. 5-3.

Each row is divided into two blocks, each with four boxes. By "energizing" these boxes, graphic characters can be created. But how does the computer know which boxes to energize? You guessed it! You have to tell the computer in such a way that it can understand. That's where the hexadecimal code comes in handy.

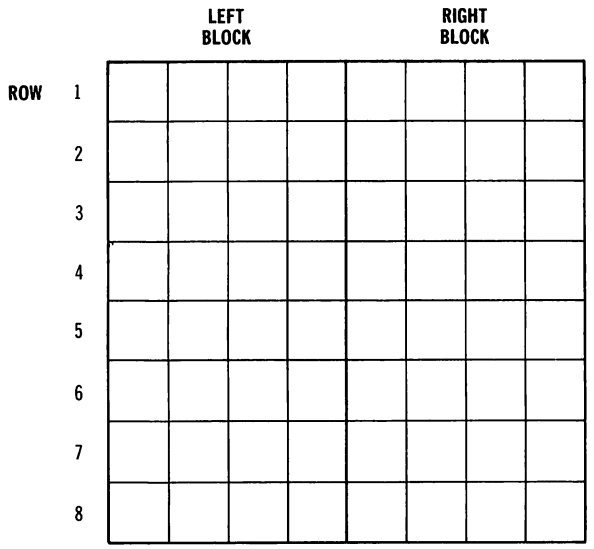


Fig. 5-3. Layout of the 64-box grid locations.

Hexadecimal Code

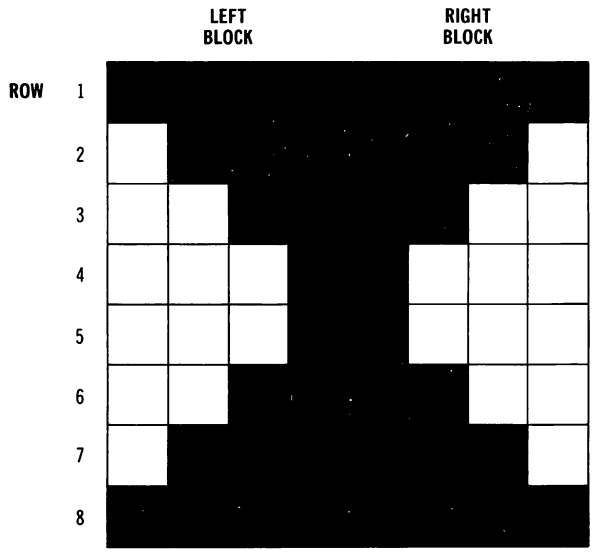
Hexadecimal code is used just like a "software light switch." The code is used to tell the computer which boxes in each block you want to energize. Fig. 5-4 shows each possible block condition and the hexadecimal code for each.

By using the illustration given in Fig. 5-4, you can fill in the boxes in each block anyway you like. Remember though that each block has four boxes, so you must tell the computer what to do with each and every block. Let's create a graphic character together and figure out the hexadecimal code. Let's begin with an easy one—a solid block.

If we want all the blocks in row 1, Left Block, left in the on position, we must give the computer the hexadecimal code for the "all on" condition. From Fig. 5-5, we see that the code for "all on" is the letter F. This tells the computer to energize all four of the boxes in row 1, Left Block. We now must tell the computer what to do with the Right Block. Since F is the code for "all on" and we want row 1, Right Block, to be "all on" also, the code is F again. In fact, for a solid black-box character, all the row and block positions must be "on," so the F code is used for each row and block.

Thus, the hexadecimal code for a solid black-box character is

FFFFFFFFFFFFFFFF (16 F's). Since there are a total of 16 blocks, each character we create must have 16 numbers or letters describing it. Let's try another character that requires a little imagination. Try and figure out the codes on your own.



If you used the block-condition code chart, you should have been able to figure this out. The code for this character is FF7E3C18183C7EFF which has sixteen numbers and letters. The following diagram shows the interpretation for the code.

ROW	LEFT BLOCK	RIGHT BLOCK	CODE FOR LEFT BLOCK	CODE FOR RIGHT BLOCK
1	XXXX	XXXX	F	F
2	XXX	XXX	7	E
3	XX	XX	3	C
4	X	X	1	8
5	X	X	1	8
6	XX	XX	3	C
7	XXX	XXX	7	E
8	XXXX	XXXX	F	F

We suggest that you use graph paper to sketch your characters and to

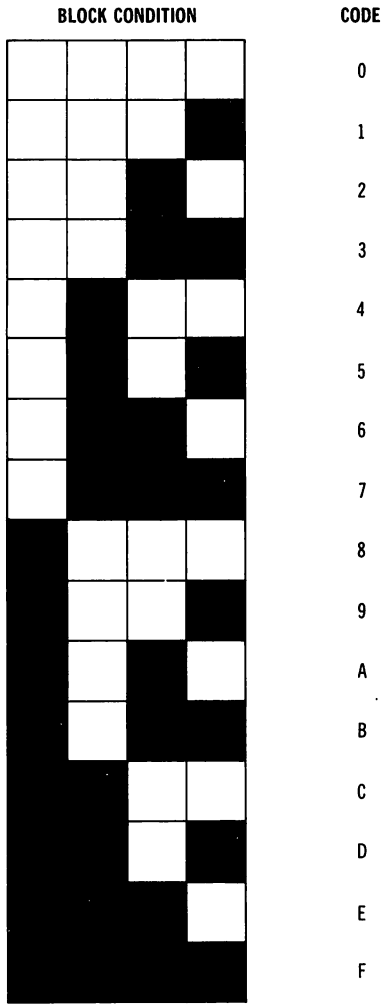


Fig. 5-4. Code position of each tiny block.

write in the code for each block condition before you enter it into a program.

Calling Your Character

After you have created your new character, you must get it into your programs. The CALL CHAR subprogram does exactly that. Let's look at the components of this subprogram. This is the way that a CALL CHAR statement for the solid box which we created earlier would look.

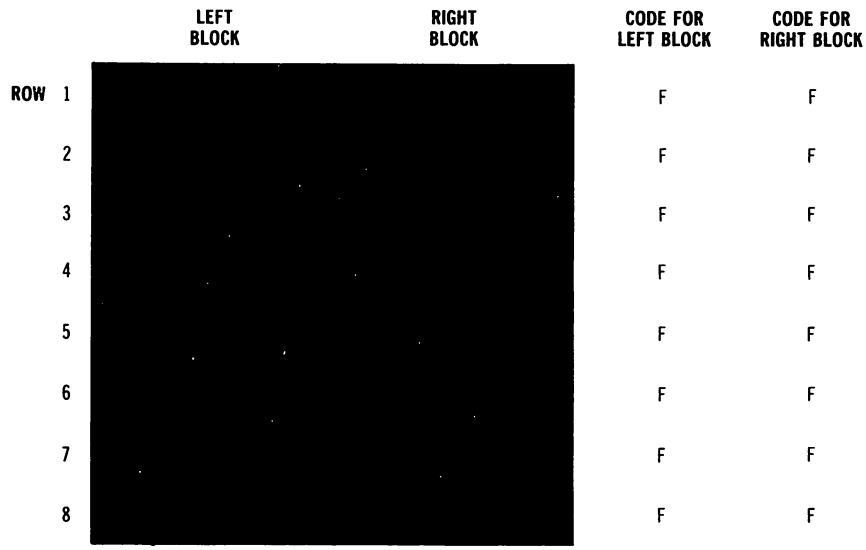


Fig. 5-5. Creating a solid block.

```
10 CALL CHAR(33,"FFFFFFFFFFFFFFFF")
```

The 16 F's are the codes for the solid-block character. They must be in quotation marks.

The first number in the parentheses is an ASCII number that you are assigning the new character. You can use any ASCII number you like, but we recommend using a little used punctuation mark. While the program is running, you will not be able to use the keyboard character that you renamed.

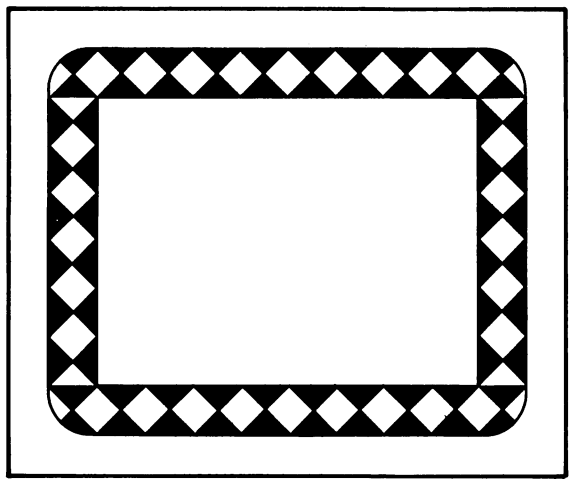
Once the CALL CHAR line is in a program, you can place the character on the screen by using either VCHAR or HCHAR. Try the following:

```
10 CALL CHAR(33,"FF7E3C18183C7EFF")
20 CALL VCHAR(10,12,33,5)
```

Your new character should appear on the screen at row 10, column 12, and it will repeat 5 times. You can put several of your characters together to draw pictures. We will show you how later.

Now that we have discussed all of the graphic subprograms, let's put them together and do some graphics.

COMBINED GRAPHICS SUBROUTINE



Type in the following program.

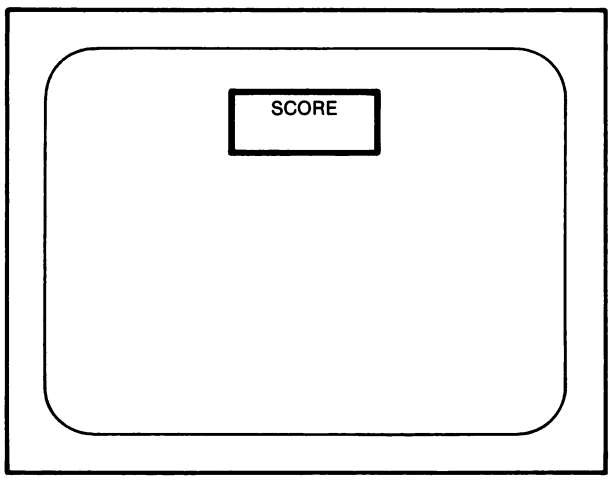
```
100 CALL CLEAR                    -Clears the screen.
110 CALL CHAR(33,"FF7E3C          -Defines the first character.
    18183C7EFF")
115 CALL CHAR(38,"FFE7C           -Defines the second character.
    38181C3E7FF")
120 CALL VCHAR(2,2,33,22)         -Beginning at R2 C2, a new char-
                                   acter is displayed vertically 22
                                   times.
130 CALL HCHAR(1,1,33,30)         -Beginning at R1 C1, a new char-
                                   acter is displayed horizontally 30
                                   times.
140 CALL VCHAR(2,30,33,22)        -Beginning at R2 C30, a new
                                   character is displayed vertically
                                   22 times.
150 CALL HCHAR(24,1,33,30)        -Beginning at R24 C1, a new
                                   character is displayed horizon-
160 GOTO 110                       tally 30 times.
```

You should see a neat diamond-like border. You can change the placement of the border by changing the row and column numbers in lines

120 to 150. Move the border around to get a feel for graphics placement. If you plot the border at row 1, column 1 and at row 1, column 32, you won't be able to see it because of the curvature of the tv screen. Here is another useful program that uses all three graphics subprograms. Input:

```
100 CALL CLEAR
110 CALL VCHAR(2,14,83)      -Displays the letter S at row 2,
                              column 14.
120 CALL VCHAR(2,15,67)      -Displays the letter C at row 2,
                              column 15.
130 CALL VCHAR(2,16,79)      -Displays the letter O at row 2,
                              column 16.
140 CALL VCHAR(2,17,82)      -Displays the letter R at row 2,
                              column 17.
150 CALL VCHAR(2,18,69)      -Displays the letter E at row 2,
                              column 18.
160 CALL CHAR(33,"FFFFFFF    -Defines 33 as a block.
    FFFFFFFF")
170 CALL HCHAR(1,13,33,7)    -Lines 170 through 200 plot the
180 CALL HCHAR(4,13,33,7)     horizontal and vertical lines that
190 CALL VCHAR(1,13,33,3)     create a score box.
200 CALL VCHAR(1,19,33,3)
210 GOTO 110
```

Note that in this program, we have placed ASCII symbols on the



screen as well as creating a new graphic character. You can place the score box anywhere you like on the screen by changing the row and column numbers. Try using a piece of graph paper to sketch the score box someplace else on the screen and then do a program to place it on the screen.

TIC-TAC-TOE

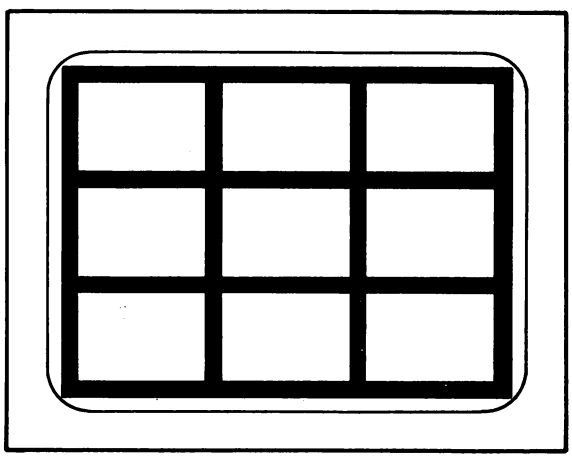
How about a TIC-TAC-TOE board? Give this a try.

```
100 CALL CLEAR
110 CALL CHAR(33,"FFFFFF
    FFFFFFFFFF")
120 CALL VCHAR(1,3,33,24)
130 CALL HCHAR(1,3,33,31)
140 CALL VCHAR(1,12,33,24)
150 CALL VCHAR(1,22,33,24)
160 CALL VCHAR(1,31,33,23)
170 CALL HCHAR(24,3,33,29)
180 CALL HCHAR(8,3,33,29)
190 CALL HCHAR(16,3,33,29)
200 GOTO 110
```

-Defines the black square.

-Lines 120 through 190 plot the black square at the given row and column coordinates.

This program displays nine squares of roughly the same size for use with games like TIC-TAC-TOE or Secret Square.



Remember when we used variables in Call Color and Call Sound? Well, we can use variables in the Graphics subprograms also. Input this:

```
100 CALL CLEAR
110 A$ = ("FF7E3C18183C7E
    FF")
120 CALL CHAR(33,A$)
130 FOR C = 1 TO 5
140 CALL VCHAR(10,C+3,33,
    4)
150 CALL VCHAR(14,C+3,33,
    4)
160 NEXT C
170 GOTO 110
```

-Defines the new character using the string symbol.

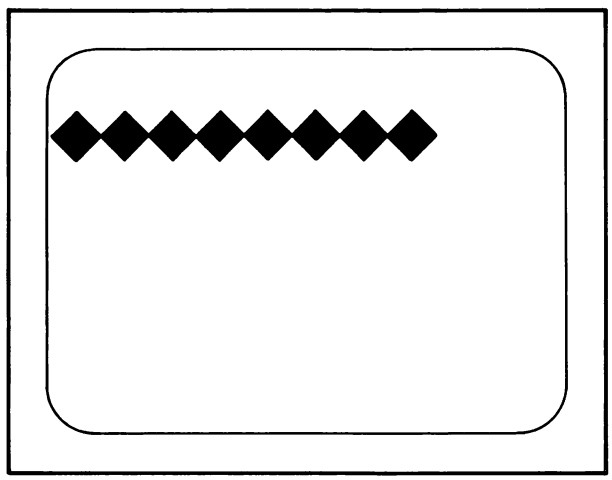
-Calls the character using the variable string defined in line 110.

-States column variable as 1 to 5.

-Lines 140 and 150 call the character at row 21, column 1 through C+3 and repeats it four times.

-Tells the computer to go to the next value of Y.

The use of variables lets you display the same character several times.



SCREEN FILLER

Here is a little program that will fill the screen with any ASCII code that you select.

```
5  CALL CLEAR
10 INPUT "CODE":CODE
```

```
20 INPUT "SECOND CODE":CODEII
30 CALL VCHAR(1,1,CODE,772)
40 CALL HCHAR(1,1,CODEII,800)
50 GOTO 10
```

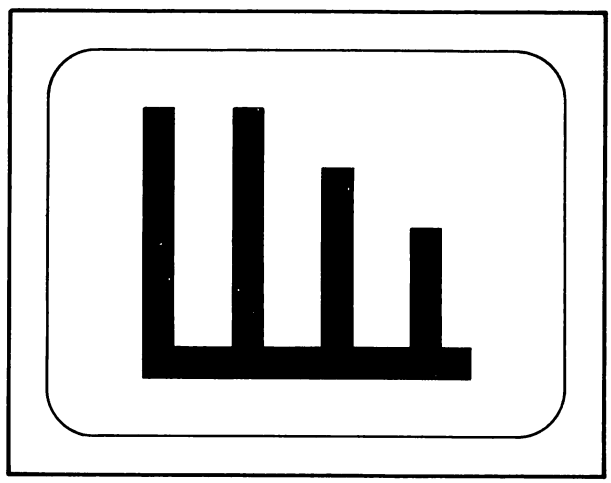
Use this program to get an idea of how a full screen of standard keyboard characters will look. You might want to use a character screen display in a game later.

BAR GRAPH

You can also plot bars on a bar graph using these subprograms. Here is one way:

```
100 CALL CLEAR
110 CALL CHAR(33,"FFFFFFFFF
    FFFFFFF")
120 CALL VCHAR(4,10,33,17)
130 CALL HCHAR(20,11,33,21)
140 CALL VCHAR(10,15,33,10)
150 CALL VCHAR(5,20,33,15)
160 CALL VCHAR(10,30,33,10)
170 GOTO 110
```

-Lines 120 and 130 display X and Y axis.

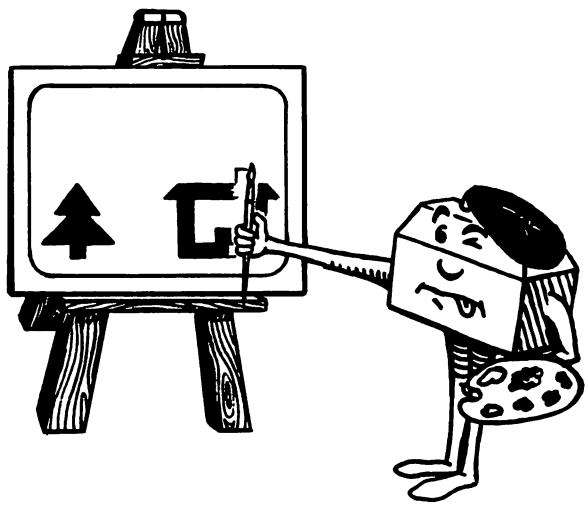


Lines 140 to 160 will display the bar graph. By changing the row and column numbers and the number of repeats, you can plot anything. Remember that the TI-99/4A computer begins a display at the row and column coordinates and repeats in a downward direction.

You can experiment as much as you like with all of these programs. When you are ready, let's look at ways that you can draw real pictures with the TI computer.

DRAWING WITH THE TI COMPUTER

We told you earlier in this chapter that you can put graphic characters, which you create, together to make pictures. All you need to do is make some preliminary sketches on graph paper first and define the characters you want to use. We suggest you use REM statements to identify your characters and the parts of your drawings.



Let's Draw a Blackbird

```
100 CALL CLEAR
110 REM BODY
120 CALL CHAR(33,"81C3E7      -Defines the body character.
    FFFFE7C381")
130 REM LEFT WING
140 CALL CHAR(34,"0103070     -Defines the left-wing character.
    F1F3F7FFF")
```

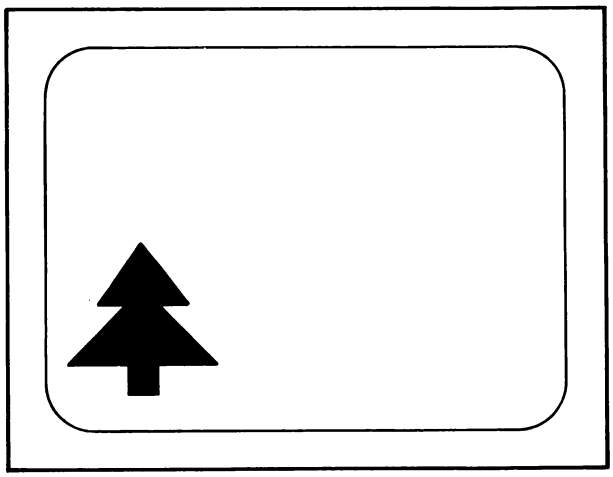
```
150 REM RIGHT WING
160 CALL CHAR(35,"80C0E0      -Defines the right-wing character.
    F0F8FCFFFF")
170 REM HEAD
180 CALL CHAR(36,"0000001     -Defines the head character.
    83C3C3C18")
190 CALL VCHAR(12,16,33)      -Displays the body.
200 CALL VCHAR(12,15,34)      -Displays the left wing.
210 CALL VCHAR(12,17,35)      -Displays the right wing.
220 CALL VCHAR(11,16,36)      -Displays the head.
230 GOTO 180
```

The preceding program is just one example of using different graphics characters to draw a picture. Try adding a second bird at another location on the screen. Or, how about adding a pine tree to the picture?

Pine Tree

```
100 CALL CLEAR
110 CALL CHAR(33,"0103070F1F3F7FFF")
120 CALL CHAR(34,"80C0E0F0F8FCFFFF")
130 CALL CHAR(35,"FFFFFFFFFFFFFFFF")
140 CALL HCHAR(17,7,33)
150 CALL HCHAR(17,8,34)
160 CALL HCHAR(18,6,33)
170 CALL HCHAR(19,6,33)
180 CALL HCHAR(19,9,34)
190 CALL HCHAR(18,7,35,2)
200 CALL HCHAR(18,9,34)
210 CALL HCHAR(19,7,35,2)
220 CALL HCHAR(20,5,33)
230 CALL HCHAR(20,6,35,4)
240 CALL HCHAR(20,10,34)
250 CALL HCHAR(21,5,35,6)
260 CALL HCHAR(21,4,33)
270 CALL HCHAR(21,11,34)
280 CALL HCHAR(22,7,35,2)
290 GOTO 110
```

This program will display a pine tree in the lower left-hand corner of the screen.



Now, let's try putting a house next to our tree.

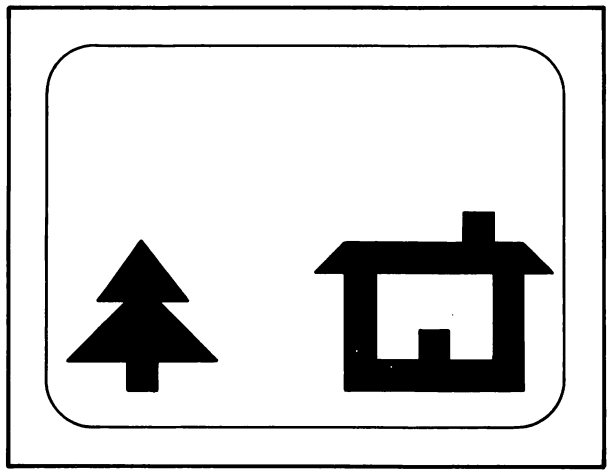
```
290 CALL HCHAR(22,18,35,10)
300 CALL HCHAR(18,18,35,10)
310 CALL HCHAR(17,19,35,8)
320 CALL HCHAR(18,17,33)
330 CALL HCHAR(18,28,34)
340 CALL VCHAR(19,18,35,3)
350 CALL VCHAR(19,27,35,3)
360 CALL HCHAR(17,18,33)
370 CALL HCHAR(17,27,34)
380 CALL HCHAR(16,26,35)
390 CALL HCHAR(21,22,35,2)
400 GOTO 110
```

Lines 290 through 400 will plot the house using the characters given at the beginning of the Pine Tree program.

Add windows and a fence to the sketch if you like, or maybe a garage. You can draw all kinds of things with the TI computer. You are limited only by your imagination.

PROGRAMS WITH COLOR, SOUND, AND GRAPHICS

Now that we have covered all three of the color, sound, and graphics areas, let's do a couple of programs that use all three subprograms.



After you have run these programs, you should be able to change them and add your own touches.

Little Cabin in the Woods

Type in the following program.

```
100 CALL CLEAR
110 CALL CHAR(33,"0103070
    F1F3F7FFF")
120 CALL CHAR(34,"80C0E0
    F0F8FCFFFF")
130 CALL CHAR(35,"FFFFFFF
    FFFFFFFFF")
140 CALL COLOR(1,16,2)

150 CALL HCHAR(17,7,33)
160 CALL HCHAR(17,8,34)
170 CALL HCHAR(18,6,33)
180 CALL HCHAR(19,6,33)
190 CALL HCHAR(19,9,34)
200 CALL HCHAR(18,7,35,2)
210 CALL HCHAR(18,9,34)
220 CALL HCHAR(19,7,35,2)
230 CALL HCHAR(20,5,33)
240 CALL HCHAR(20,6,35,4)
250 CALL HCHAR(20,10,34)
```

-Lines 110 through 130 define the character we will use to build the cabin and the tree.

-Calls white characters on a black background.

-Lines 150 through 290 display the tree.

```
260 CALL HCHAR(21,5,35,6)
270 CALL HCHAR(21,4,33)
280 CALL HCHAR(21,11,34)
290 CALL HCHAR(22,7,35,2)
300 CALL HCHAR(22,18,35,
    10)
310 CALL HCHAR(18,18,35,
    10)
320 CALL HCHAR(17,19,35,8)
330 CALL HCHAR(18,17,33)
340 CALL HCHAR(18,28,34)
350 CALL VCHAR(19,18,35,3)
360 CALL VCHAR(19,27,35,3)
370 CALL HCHAR(17,18,33)
380 CALL HCHAR(17,27,34)
390 CALL HCHAR(16,26,34)
400 CALL VCHAR(21,22,35,2)
```

-Lines 300 through 400 display the house.

```
405 CALL COLOR(2,16,2)
```

-Colors the stars white on a black background.

```
410 RANDOMIZE
```

-Tells the computer to use random numbers.

```
420 ROW = INT(10*RND)+1
430 COLUMN = INT(32*RN
    D)+1
```

-Lines 420 and 430 state variables for random placement of stars.

```
440 CALL VCHAR(ROW,COL
    UMN,42)
```

-Calls the stars.

```
450 IF F = 999 THEN 410
```

-Tells the computer to go to the star placement line when data 999 is read.

```
460 READ F
```

-Tells the TI-99/4A to read frequency from data files.

```
470 CALL SOUND(500,F,1)
```

-Calls the sounds.

```
480 GOTO 450
```

-Tells the TI-99/4A to check for 999 by going to line 450.

```
490 GOTO 490
```

-Hold loop.

```
500 DATA 262,262,392,392,4
    40
510 DATA 440,392,349,349,3
    30
520 DATA 330,294,294,262,3
    92
```

-Lines 500 to 580 state the frequencies that are used to play Twinkle, Twinkle, Little Star.

```
530 DATA 392,349,349,330,3
    30
540 DATA 294,392,392,349,3
    49
550 DATA 330,330,294,262,2
    62
560 DATA 392,392,440,440,3
    92
570 DATA 349,349,330,330,2
    94
580 DATA 294,262,999
```

What changes can you make to this program? Change the song? How about a church instead of a house and a hymn instead of Twinkle, Twinkle, Little Star? Maybe a graveyard with a scary song?

Now that you understand how each program line works, you should be able to change them.

Graphic Pattern

Let's try a graphic pattern now.

```
100 CALL CLEAR
110 FOR X = 1 TO 16
120 CALL COLOR(5,X,14)
130 CALL COLOR(8,12,X)
140 CALL COLOR(1,X,12)
150 CALL SOUND(500,-2,0)
160 CALL SOUND(500,-1,0)
170 CALL CHAR(33,"FFFFFFF
    FFFFFFFFF")
180 CALL HCHAR(5,15,33,2)
190 CALL HCHAR(6,14,33,4)
200 CALL HCHAR(7,13,33,6)
210 CALL HCHAR(8,12,33,8)
220 CALL HCHAR(9,11,33,10)
230 CALL HCHAR(10,10,33,1
    2)
240 CALL HCHAR(11,9,33,14)
250 CALL HCHAR(12,8,33,16)
260 CALL HCHAR(13,9,33,14)
270 CALL HCHAR(14,10,33,1
    2)
```

-States the color variable.

-Lines 120 through 140 call the colors for all the characters used.

-Lines 150 and 160 call the two buzzer sound signals.

-Defines the square box character.

-Lines 180 to 320 display the large diamond in the center of the screen.

```
280 CALL HCHAR(15,11,33,1
    0)
290 CALL HCHAR(16,12,33,8)
300 CALL HCHAR(17,13,33,6)
310 CALL HCHAR(18,14,33,4)
320 CALL HCHAR(19,15,33,2)
330 CALL CHAR(66,"FF7E3C      -Defines the diamond border.
    18183C7EFF")
340 CALL HCHAR(3,5,66,22)     -Lines 340 to 370 display the
350 CALL HCHAR(23,5,66,22)     border.
360 CALL VCHAR(4,5,66,20)
370 CALL VCHAR(4,26,66,20)
380 CALL CHAR(88,"AA55A       -Defines checkerboard character.
    A55AA55AA55")
390 CALL HCHAR(12,15,88,3)    -Lines 390 and 400 display the
400 CALL HCHAR(13,15,88,3)     checkerboard.
410 NEXT X                    -Tells the computer to flash the
420 GOTO 110                   next value of X.
```

Just as before, this program can easily be changed. Try displaying a square or even a circle. Or, display two or three diamonds inside the border.

The more you experiment with graphics, the better you will get at it. The graphics subprograms discussed in this chapter will be used over and over again as we continue on to the next chapter. However, if you think working with graphics was fun, wait until we make them come "alive" through the use of animation.

Numbers and Symbols

Here are some programs that can create some useful number graphics. When we deal with math games later, you will be able to use them. These programs will display block numerals 0 through 9.

Number 1

```
10 CALL CLEAR
20 CALL CHAR(33,"FFFFFFFFFFFFFFFF")
30 CALL VCHAR(12,16,33,6)
40 GOTO 30
```

Number 2

```
10 CALL CLEAR
20 CALL CHAR(69,"FEFCF8F0E0C08000")
30 CALL CHAR(33,"FFFFFFFFFFFFFFFF")
40 CALL CHAR(37,"000103070F1F3F7F")
50 CALL VCHAR(14,18,69)
60 CALL VCHAR(15,17,69)
70 CALL VCHAR(16,16,69)
80 CALL VCHAR(14,17,37)
90 CALL VCHAR(15,16,37)
100 CALL VCHAR(16,15,37)
110 CALL HCHAR(12,15,33,4)
120 CALL HCHAR(13,18,33)
130 CALL HCHAR(17,15,33)
140 CALL HCHAR(18,15,33,4)
150 GOTO 20
```

Number 3

```
10 CALL CLEAR
20 CALL CHAR(33,"FFFFFFFFFFFFFFFF")
30 CALL HCHAR(10,15,33,4)
40 CALL VCHAR(10,18,33,6)
50 CALL HCHAR(16,15,33,4)
60 CALL HCHAR(13,16,33,3)
70 GOTO 20
```

Number 4

```
10 CALL CLEAR
20 CALL CHAR (33, "FFFFFFFFFFFFFFFF")
30 CALL VCHAR (10,15,33,4)
40 CALL VCHAR (10,18,33,6)
50 CALL HCHAR (13,16,33,2)
60 GOTO 20
```

Number 5

```
10 CALL CLEAR
20 CALL CHAR (33,"FFFFFFFFFFFFFFFF")
30 CALL HCHAR (10,15,33,4)
40 CALL HCHAR (11,15,33)
```

```
50 CALL HCHAR (12,15,33,4)
60 CALL VCHAR (12, 18,33,4)
70 CALL HCHAR (15,15,33,3)
80 GOTO 20
```

Number 6

```
10 CALL CLEAR
20 CALL CHAR (33,"FFFFFFFFFFFFFFFF")
30 CALL VCHAR (10,15,33,6)
40 CALL HCHAR (15,16,33,2)
50 CALL VCHAR (13,17,33,3)
60 CALL VCHAR (13,16,33)
70 GOTO 20
```

Number 7

```
10 CALL CLEAR
20 CALL CHAR (33,"FFFFFFFFFFFFFFFF")
30 CALL HCHAR (10,15,33,4)
40 CALL VCHAR (11,18,33,6)
50 GOTO 20
```

Number 8

```
10 CALL CLEAR
20 CALL CHAR (33,"FFFFFFFFFFFFFFFF")
30 CALL HCHAR (10,15,33,3)
40 CALL HCHAR (14,15,33,3)
50 CALL VCHAR (11,15,33,3)
60 CALL VCHAR (11,17,33,3)
70 CALL VCHAR (12,16,33)
80 GOTO 20
```

Number 9

```
10 CALL CLEAR
20 CALL CHAR (33,"FFFFFFFFFFFFFFFF")
30 CALL HCHAR (10,15,33,3)
40 CALL VCHAR (11,15,33,2)
50 CALL HCHAR (12,16,33,2)
60 CALL VCHAR (11,17,33,5)
70 GOTO 20
```

Number 0

```
10 CALL CLEAR
20 CALL CHAR(33,"FFFFFFFFFFFFFFFF")
30 CALL HCHAR(10,15,33,3)
40 CALL VCHAR(11,15,33,4)
50 CALL VCHAR(11,17,33,4)
60 CALL HCHAR(15,15,33,3)
70 GOTO 20
```

Plus Sign in Middle of Screen

```
10 CALL CLEAR
20 CALL CHAR(33,"FFFFFFFFFFFFFFFF")
30 CALL HCHAR(12,16,33)
40 CALL HCHAR(13,15,33,3)
50 CALL HCHAR(14,16,33)
60 GOTO 20
```

Minus Sign

Delete lines 30 and 50 from the preceding subroutine and you will get a minus sign.

Division Sign

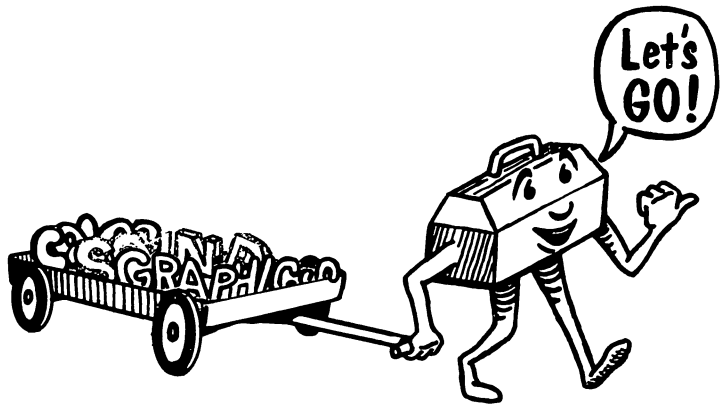
```
10 CALL CLEAR
20 CALL CHAR(33,"FFFFFFFFFFFFFFFF")
30 CALL HCHAR(13,15,33,3)
40 CALL CHAR(69,"00183C3C18000000")
50 CALL HCHAR(12,16,69)
60 CALL HCHAR(14,16,69)
70 GOTO 20
```

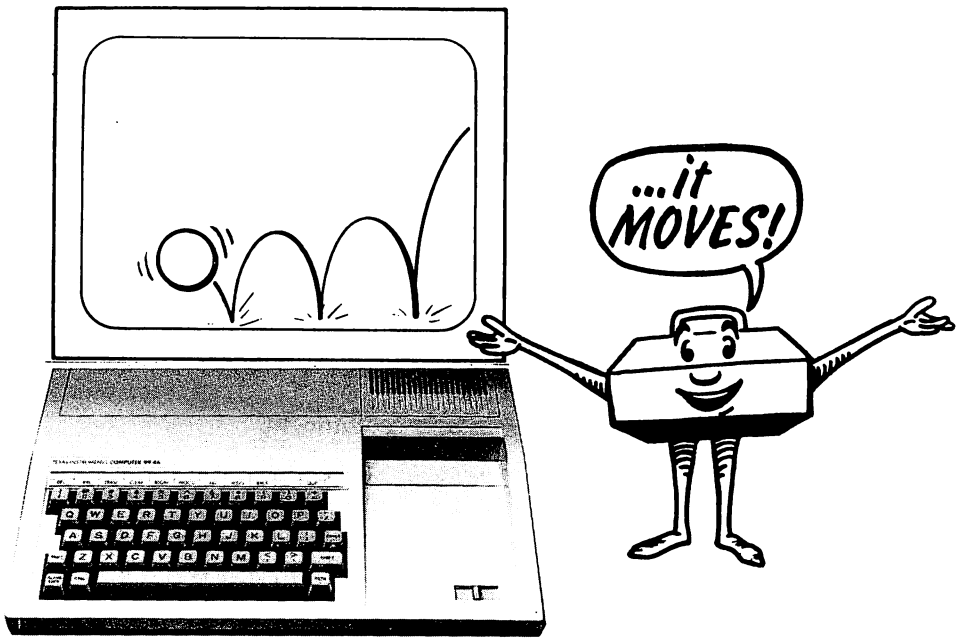
Multiplication Sign

```
10 CALL CLEAR
20 CALL CHAR(33,"FFFFFFFFFFFFFFFF")
30 CALL HCHAR(12,16,33)
40 CALL HCHAR(11,15,33)
50 CALL HCHAR(11,17,33)
60 CALL HCHAR(13,15,33)
```

```
70 CALL HCHAR (13,17,33)
80 GOTO 20
```

In this chapter, we have discussed all of the graphic subprograms and how to use them. We suggest that you make sure that you understand this chapter completely before moving on to further features. In Chapter 6, we will show you how to make your graphics figures move.





Animation Subroutines 6

Animation, or moving graphics characters around on the screen, is a lot of fun. And, although it may seem difficult, it really isn't. Once you have mastered the graphics subprograms that we discussed in Chapter 5, you are ready to animate.

It is helpful, however, if you fully understand how the computer animates. Animation is created by first plotting a character on the screen in one place, erasing it, and then plotting the character again in another place. The computer does this three-step operation so extremely quick that all we see is the image (the character) blinking along. We can't really see the erasing process; it happens too fast.

ANIMATION IN ONE SCREEN LOCATION

We have used the FOR NEXT loop quite a bit in earlier chapters. This loop is one way that the TI computer draws and, then, redraws a character. Let's try a very simple animation program. Input the following:

```
100 CALL CLEAR                -Clears the screen.
110 FOR X = 32 TO 47          -Defines X as ASCII characters 32
                               to 47.
120 CALL HCHAR(12,10,X)       -Places the character at row 12,
                               column 10.
130 NEXT X                    -Tells the computer to go back
                               and get the next value of X so
                               that line 120 can display it.
```

On the first cycle of the loop, this program displays ASCII character 32, which is the space character. You won't be able to see the "space" move. The next cycle will display the exclamation point and so on, through ASCII number 47 (the slant bar). Since only one character can occupy a screen location at a time, each new value of X "erases" the old one. By using this technique, you can give an illusion of movement. Let's try something a little more fun than flashing exclamation points and, at the same time, another way to do animation within one screen location.

Jumping Jack

Let's make a Jumping Jack program.

```
100 CALL CLEAR
110 CALL CHAR(33,"18183CAA18182424")
```

```
120 CALL CHAR(34,"18AA3C1818244200")
130 CALL VCHAR(12,16,33)
140 CALL VCHAR(12,16,34)
150 GOTO 110
```

When you type RUN and ENTER, you should see a little stick figure jumping up and down at row 12, column 16.

Let's look carefully at this program. Lines 110 and 120 define the characters illustrated in Figs. 6-1 and 6-2. The little man in Fig. 6-1 has his hands to his sides and his legs are straight. But the man shown in Fig. 6-2 has his arms over his head and his feet spread apart. Line 130 displays character 33, and line 140 displays character 34, at row 12, column 16. We know that only one character can occupy a screen location at any one time. So lines 130 and 140 display their respective characters time after time. However, if line 150 is left out, the character will be displayed only once.

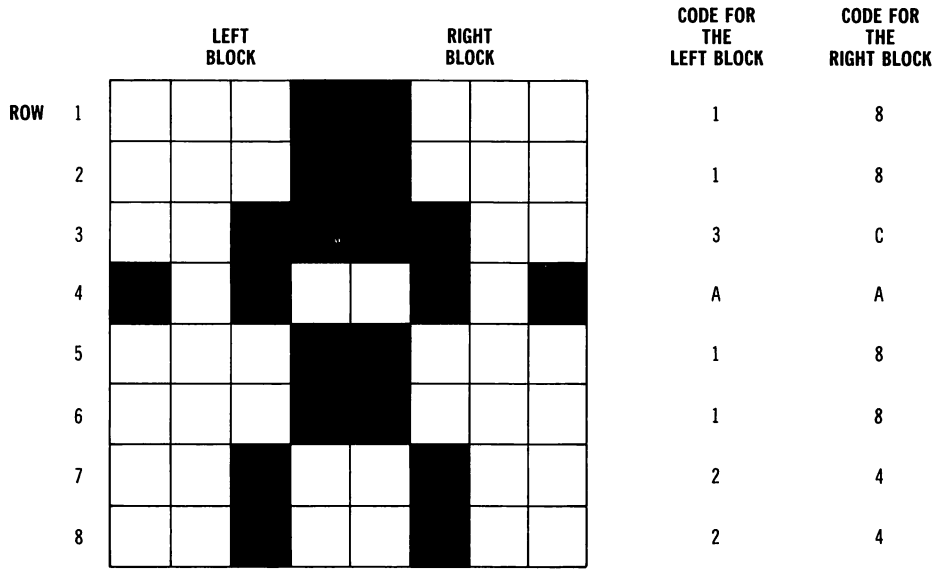


Fig. 6-1. Hex grid locations for the figure at rest.

Whether we use FOR NEXT loops or character definitions with two or more VCHAR or HCHAR commands, the result is the same. If you are dealing with animating just one screen location, you are simply drawing and redrawing the characters over and over in that location.

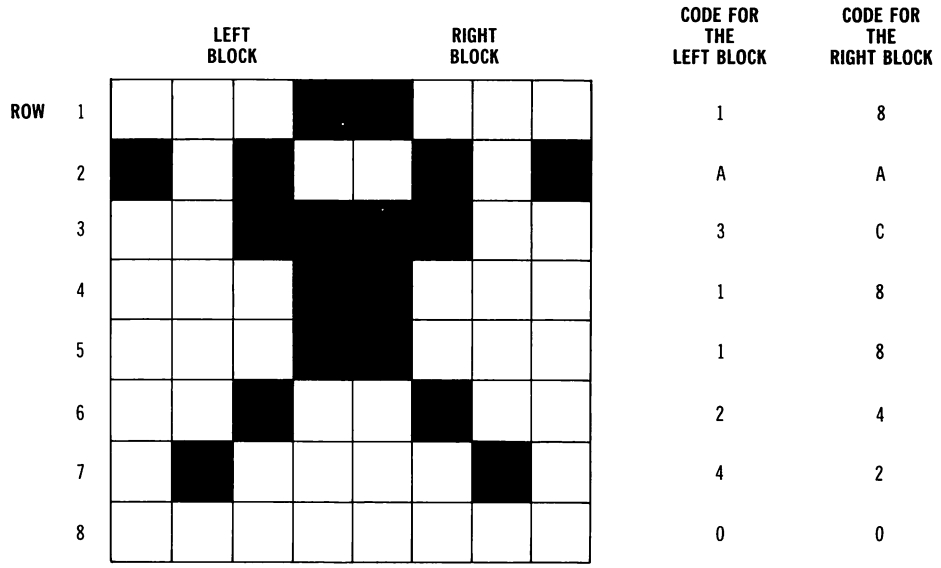


Fig. 6-2. Hex grid locations for the jumping figure.

HORIZONTAL ANIMATION

By using a FOR NEXT loop, we can make our character move from one screen location to another.

Let's use our little jumping man character and move him across the screen (Fig. 6-3). Type in the following:

```
100 CALL CLEAR
110 CALL CHAR(33,"18183CAA18182424")
120 FOR X = 1 TO 32
130 CALL HCHAR(10,X,33)
140 NEXT X
```

When you type RUN and ENTER, you will see that the little man moves across the screen from left to right, but you will get a string of little men, not just one walking along. That is because we didn't tell the computer to erase any figures, so it didn't.

To get just one man walking across the screen, we need to add a CALL CLEAR line to the program. Add the following:

```
135 CALL CLEAR
```

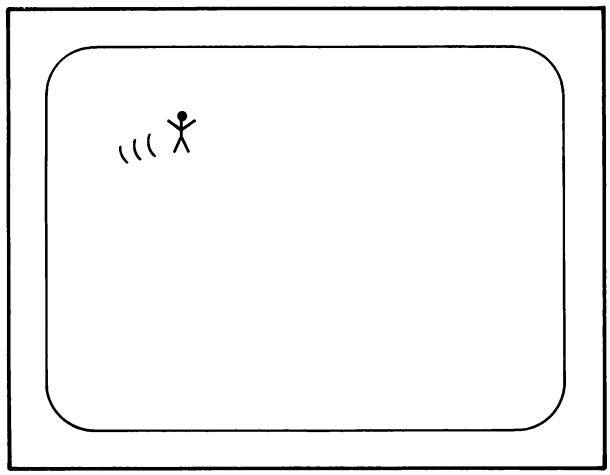


Fig. 6-3. Screen display of Jumping Jack moving across the screen.

Now when the program runs, you will see just one figure moving across the screen. The CALL CLEAR line erases the figure displayed by line 130 before the computer reads the next value of X.

Now that we know how to get the character to move from left to right, let's see if we can get it to come back across from right to left. All we need to do is change line 120. Edit line 120 so that it looks like this:

```
120 FOR X = 32 TO 2 STEP -1
```

(Note that you must use a 2 instead of a 1, because 2 − 1 is 1.)

This change tells the computer that the first value of X is now 32, not 1. The STEP −1 command tells the computer to move at intervals of 1 in a *minus* direction. In other words, the first location of our little man is row 10, column 32, and the next location is row 10, column 31. The computer will continue to subtract 1 from the last screen location and will move all the way across the screen. Now that we know how to move a character from right to left and from left to right, let's see if we can put the two directions together. This new program will send our little friend over to the right side of the screen and, then, will bring him back again. Input:

```
100 CALL CLEAR
110 CALL CHAR(33,"18183CAA18182424")
120 FOR X = 1 TO 32
```

```
130 CALL HCHAR(10,X,33)
140 CALL CLEAR
150 NEXT X
160 FOR X = 32 TO 2 STEP-1
170 CALL HCHAR(10,X,33)
180 CALL CLEAR
190 NEXT X
```

Not bad, huh? All we did was combine two FOR NEXT loops together. Lines 130 to 150 display the figure going from left to right and lines 160 to 190 display it moving from right to left.

If you have been watching the screen, you will notice that the figure is not jumping. That is because we haven't put in character number 34 or the little man's jumping condition. Let's make our little friend really jump. Try this:

```
100 CALL CLEAR
110 CALL CHAR (33,"18183C      -Defines character at rest.
    AA18182442")
120 CALL CHAR (34,"18AAE       -Defines character jumping.
    C1818244200")
130 FOR X = 1 TO 31            -Defines the value of X which
                                makes the character move to
                                the right.
140 Y =X + 1                   -Defines the column for the jump-
                                ing state.
150 CALL HCHAR (10,X,33)       -Displays the character at rest.
160 CALL CLEAR                 -Erases the character at rest.
170 CALL HCHAR (10,Y,34)       -Displays character jumping.
180 CALL CLEAR                 -Erases the character jumping.
190 NEXT X                     -Loops back for next column.
200 FOR X = 32 TO 2 STEP       -Lines 200 through 260 move the
    -1                          character from the right side of
210 Y = X - 1                   the screen back to the left side.
220 CALL HCHAR (10,X,33)
230 CALL CLEAR
240 CALL HCHAR (10,Y,34)
250 CALL CLEAR
260 NEXT X
```

If you have entered the program correctly, you should see the little man actually jump across the screen. Beginning at row 10, column 1, the

little man is at rest. At row 10, column 2 (or X + 1), the little man jumps; this loop continues all the way across the screen. There are other ways that you can make the little man move. By changing the row number in lines 170 and 240, you can make the little man jump up or down. Change 170 to read:

```
170 CALL HCHAR (8,Y,34)
```

and line 240 to read:

```
240 CALL HCHAR (8,Y,34)
```

He really goes now, doesn't he? Try defining your own characters and experiment with this program. Flying saucers can be fun, too, or how about a car?

VERTICAL ANIMATION

The TI-99/4A computer can animate vertically also. The process is pretty much the same except that your FOR NEXT loops work with row numbers instead of column numbers.

Let's put our little man to work again. Try the following program.

```
100 CALL CLEAR
110 CALL CHAR (33,"18183C      -Defines the character at rest.
    AA18182442")
120 CALL CHAR (34,"422418      -Defines the character falling.
    18AA3C1818")
130 FOR X = 24 TO 1 STEP       -Defines the row numbers going
    -1                          up.
140 CALL VCHAR (X,16,33)       -Displays the character at rest.
150 CALL CLEAR                 -Erases the character at rest.
160 NEXT X                     -Loops back for next row.
170 FOR X = 1 TO 24            -Defines the row numbers com-
                                ing down.
180 CALL VCHAR (X,16,34)       -Displays man falling head first.
190 CALL CLEAR                 -Erases the falling man so that we
                                can display him again, one posi-
                                tion further down on the screen.
200 NEXT X                     -Gets the next row number.
```

Lines 170 to 200 are used to bring our friend back down from the top of the screen. See how it works? To move a character up the screen, simply

tell the computer where you want it to start (in this case, row 24) and where you want it to stop (row 1 in this program). Don't forget that to make the character climb, you must include a step on a negative number.

Jack Saved by "IF THEN"

In our last program, we let the little man fall to his death, head first. We can save his life with an IF/THEN statement. In this next program, we will tell the computer to give our little friend a parachute (Fig. 6-4) when he reaches a certain altitude. We are really telling the computer to test both for a certain condition, or a value of X, and what to do when it finds that condition. Input this:

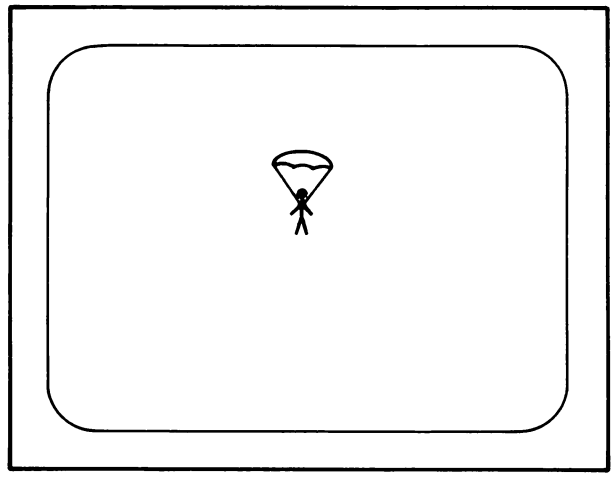


Fig. 6-4. Jumping Jack has a parachute.

```
100 CALL CLEAR
110 CALL CHAR(33,"18183C      -Defines man.
    AA18182442")
120 CALL CHAR(34,"422418      -Defines falling man.
    18AA3C1818")
130 CALL CHAR(35,"327EAA      -Defines parachute.
    8142241818")
140 FOR X = 24 TO 1 STEP      -Lines 140 to 170 display the man
    -1                         going up.
```

```
150 CALL VCHAR (X,16,33)
160 CALL CLEAR
170 NEXT X
180 FOR X = 1 TO 24
190 CALL VCHAR (X,16,34)
200 CALL CLEAR
210 IF X = 10 THEN 230
220 NEXT X
230 FOR X = 9 TO 24
240 Y = X - 1
250 CALL VCHAR(X,15,33)
270 CALL VCHAR(Y,15,35)
280 CALL CLEAR
290 NEXT X
```

-Lines 180 to 200 display the man falling.

-This tells the computer that when X = 10, it is to go to line 230. This will save the little man's life.

-Lines 230 to 290 turns the man rightside up and opens his parachute. Line 210 is the life-saving line.

We will use IF/THEN statements in the rest of the book. This statement is an excellent way of "branching" the program when a certain condition exists. We will discuss the IF/THEN statement in more detail later in this chapter.

Now that we know how to animate vertically and horizontally, we are ready to experiment with some interesting animation programs. All of the programs that we discuss can be changed to take care of practically any animation need you might have. The basic animation subroutines we are about to show you will be used a great deal when we begin building games in Chapter 9.

FUN WITH ANIMATION

Everyone knows how popular PAC MAN is. Well, the TI-99/4A computer can make a similar "critter" with the following little program.

```
100 CALL CLEAR
110 CALL COLOR(1,11,2)
120 CALL COLOR(5,10,2)
130 CALL CHAR(33,"327EF7F
    CF8FE7F3E")
140 CALL CHAR(34,"3C7EF7
    FFFFFF7F3E")
```

-Yellow on black for TI MAN.
-Red on black for the monster.
-Defines open-mouthed TI MAN.
-Defines closed-mouth TI MAN.

```
150 CALL CHAR(66,"183C66
    7E7E7E7E55")          -Defines red monster.
160 FOR X = 1 TO 27       -Value of column number for
                           open-mouthed TI MAN.
170 A = X + 5             -Value of column for the mon-
                           ster.
180 B = X + 1             -Value of the column for closed-
                           mouth TI MAN.
190 CALL HCHAR(12,X,33)   -Displays open-mouthed TI MAN.
200 CALL CLEAR
210 CALL HCHAR(12,B,34)   -Displays closed-mouth TI MAN.
220 CALL CLEAR
230 CALL HCHAR(12,A,66)   -Displays the monster.
240 NEXT X
```

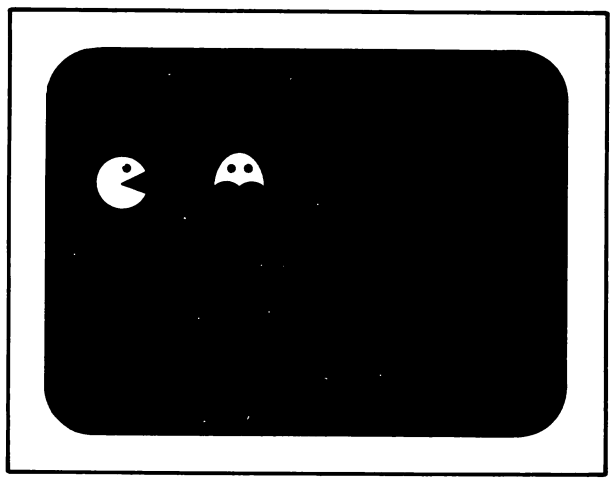


Fig. 6-5. Yellow TI MAN chasing a red monster.

Lines 160 to 240 move the TI MAN and the monster across the screen. The monster will always be 5 spaces ahead of TI MAN because of line 170 (Let A = X + 5).

It is neat to watch the chase, but TI MAN will never catch the monster. Let's make some changes and see what happens.

```
100 CALL CLEAR
110 CALL COLOR(1,11,2)    -Color for TI MAN.
120 CALL COLOR(5,10,2)    -Color for the monster.
```

```
130 CALL CHAR(33,"327EF7F
    CF8FE7F3E")
140 CALL CHAR(34,"3C7EF7
    FFFFFF7F3E")
150 CALL CHAR(66,"183C66
    7E7E7E7E55")
160 CALL SOUND(4000,-4,1)
170 FOR X = 1 TO 27
180 A = X + 5
190 B = X + 1
200 CALL HCHAR(12,X,33)

210 CALL KEY(0,KEY,R)
220 IF R = 0 THEN 240
230 IF R = 1 THEN 300
240 CALL CLEAR
250 CALL HCHAR(12,B,34)
260 CALL CLEAR
270 CALL HCHAR(12,A,66)
280 NEXT X
290 GOTO 160
300 FOR X = B TO 27 STEP 2

310 CALL HCHAR(12,X,33)
320 CALL CLEAR
330 NEXT X
340 IF X = A THEN 350

350 PRINT "YOU GOT HIM"
```

-Lines 130 and 140 define the TI MAN characters.

-Defines the monster character.

-Calls the munching noise.

-Lines 170 to 190 state the variables.

-This plots TI MAN with his mouth open.

-Lines 210 to 230 allow you to input a key for control.

-Lines 240 to 280 plot the monster and TI MAN with his mouth closed.

-Value of X is whatever the column number (B) is at the time the key is pressed, up to 27.

-Lines 300 to 330 speed up TI MAN so that he can catch the monster.

-If X = A, then TI MAN has eaten the monster.

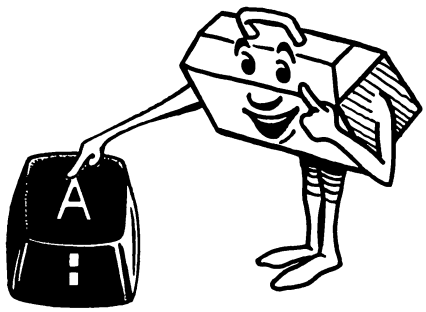
-Screen prompt for the condition in line 340.

This program is similar to the earlier TI MAN program except, in this one, you control TI MAN's speed so that he can catch the monster. This is done by using the CALL KEY subprogram (lines 210–230) and an IF/THEN statement (line 340). Let's look at the CALL KEY subprogram.

KEY SUBPROGRAM

The KEY subprogram allows you to use a character from the keyboard in a program, without using an input statement. This gives you a

chance to talk to the computer while the program is running. We can use the KEY subprogram to control movement on the screen or for several other conditions. The KEY subprogram has three variables that you need to understand. The first variable is called the Key Unit.



Key Unit

There are 5 Key Unit variables.

- **0** Activates the entire keyboard. Hitting any key will trigger the condition specified in the program.
- **1** Activates only the left side of the keyboard.
- **2** Activates only the right side of the keyboard.
- **3** Places the computer in the standard TI-99/4A keyboard mode.
- **4** Puts the keyboard in the Pascal format.
- **5** Puts the keyboard in the BASIC mode.

We will use 0, 1, and 2 quite frequently in later programs. Using 0 will allow you to control direction or speed by hitting any key. By using Key Units 1 and 2, you can control two conditions.

Return Variable

The second variable in the KEY subprogram is called the *Return Variable*. This variable must be a numeric value.

If the Key Unit used is 0, then the Return Variable should be an ASCII number between 1 and 127. If Key Unit 1 or 22 is used, an ASCII number between 0 and 19 should be used. You can also use just the word KEY to represent this variable. The computer puts the value of the last key pressed into the Return Variable place in the formula so that it can test for the last variable in the subprogram—the status variable.

Status Variable

The status variable is a numeric value that the computer assigns to the status of the keyboard input.

+1 = A new key was pressed since the last execution of the CALL KEY routine.
−1 = The same key was pressed since the last execution of the CALL KEY routine.
0 = No key was pressed since the last execution of the CALL KEY routine.

You must define these three conditions in your program and tell the computer what to do with each condition. If you use 0 as a Key Unit, you only need to define +1 and 0 in the program. Look at line 210 in our TI MAN program.

```
210 CALL KEY(0,KEY,R)
```

The 0 is the Key Unit. In this case, any key we press will work, because 0 activates the entire keyboard.

The KEY in the parentheses stands for the Return Variable. When we press a key, the ASCII code is stored in this position.

The R is the Status Variable. We defined the conditions of R in lines 220 and 230. Line 220 tells the computer to go to line 240 if no key is pressed and CALL CLEAR. But line 230 tells the computer to go to line 300 if any key is pressed. Line 300 begins the subroutine for displaying TI MAN in his faster chase mode.

We will use CALL KEY later to control animation and we will review the subprogram again when we use it.

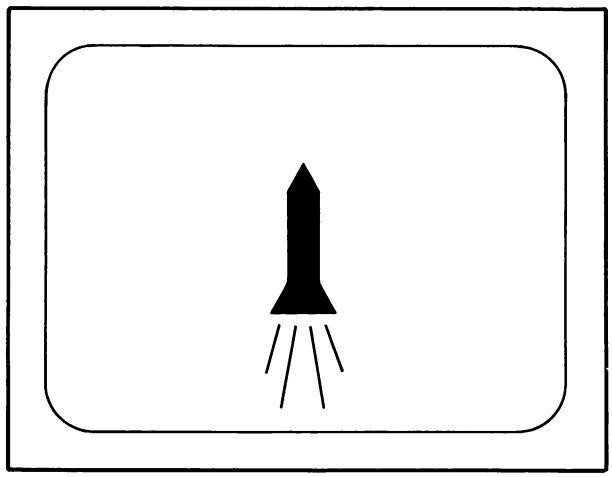
The IF/THEN Statement

The IF/THEN statement allows you to test for a specific condition and send the program in another direction or branch according to that condition. In the TI MAN program, we used an IF/THEN statement to tell us when TI MAN and the monster are in the same screen location. When TI MAN and the monster are in the same screen location, the computer will execute line 350.

We will use IF/THEN statements quite a lot later on to branch our programs when we play games. Now let's try some more animated programs.

ROCKET SHIP

Here is another program that uses the IF/THEN statement.

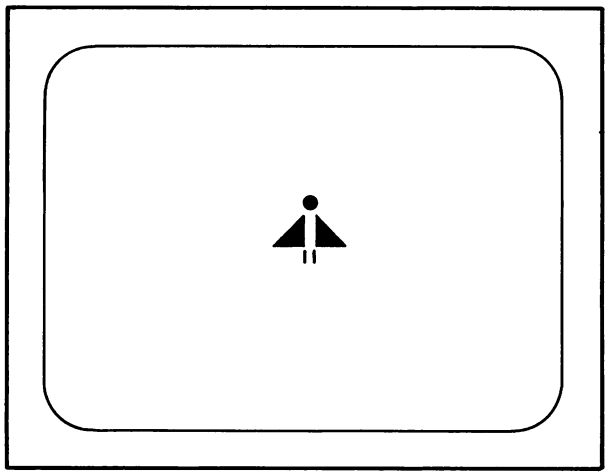


```
100 CALL CLEAR
110 CALL CHAR(33,"1038383      -Defines the rocket.
    87CFE3800")
120 FOR X = 24 TO 1 STEP       -Defines the value of X.
    -1
130 IF X = 15 THEN 170         -Branches program to line 170
140 CALL CLEAR                  when X or column =15.
150 CALL VCHAR(X,16,33)        -Lines 150 and 160 display the
160 NEXT X                      rocket.
170 CALL COLOR(1,7,12)         -Calls yellow background and
                                red rocket.
180 CALL SOUND(1000,-6,1)      -Makes the crash sound.
190 CALL CLEAR
```

Notice that when the row number equals 15, the IF/THEN statement tells the computer to go directly to the explosion section of the program.

BLACK BIRD

Here is a simple little program that we call Black Bird.



```
100 CALL CLEAR
110 REM BODY
120 CALL CHAR(33,"81C3E7FFFFE7C381")
130 REM LEFT WING
140 CALL CHAR(34,"0103070F1F3F7FFF")
150 REM RIGHT WING
160 CALL CHAR(35,"80C0E0F0F8FCFEFF")
170 REM HEAD
180 CALL CHAR(36,"000000183C3C1800")
190 REM RIGHT FLIGHT
200 CALL CHAR(40,"FFFEFCF8F0E0C080")
210 REM LEFT FLIGHT
220 CALL CHAR(41,"FF7F3F1F0F070301")
230 REM FEET
240 CALL CHAR(42,"2424240000000000")
250 CALL VCHAR(12,16,33)
260 CALL VCHAR(12,15,34)
270 CALL VCHAR(12,17,35)
```

```
280 CALL VCHAR(13,16,42)
290 CALL VCHAR(11,16,36)
300 CALL VCHAR(12,17,40)
310 CALL VCHAR(12,15,41)
320 GOTO 250
```

Lines 120 through 240 define all the bird's body parts. Lines 250 to 290 display the bird's body parts. Lines 300 and 310 displayed over lines 260 and 270 tend to cause the illusion of movement.

We used the Black Bird in the graphics chapter. Now, by defining characters for more body parts and displaying them, we can watch the bird try to fly.

SHOOTING PROGRAM

This program will fire a square block from the left of the screen to the right of the screen when you hit any key on the left side of the keyboard.

```
100 CALL CLEAR
110 CALL CHAR(33,"FFFFFFF     -Defines the black bullet block.
    FFFFFFFF")
120 CALL KEY(1,KEY,R)         -Activates the left side of the key-
                               board.
130 IF R = 1 THEN 150         -Lines 130 and 140 define the Sta-
                               tus Variables. If no key is pushed,
                               nothing happens. If any key on
140 IF R = 0 THEN 100          the left side of the keyboard is
150 FOR X = 1 TO 32 STEP 4     pushed, then the computer goes
160 CALL CLEAR                 to line 150 and fires a bullet.
170 CALL HCHAR(10,X,33)       -Plots the bullet.
180 NEXT X
190 GOTO 100
```

This subroutine will be useful if you want to shoot at something in a game. You can also shoot vertically by changing line 150 to

```
150 FOR X = 24 TO 1 STEP-1
```

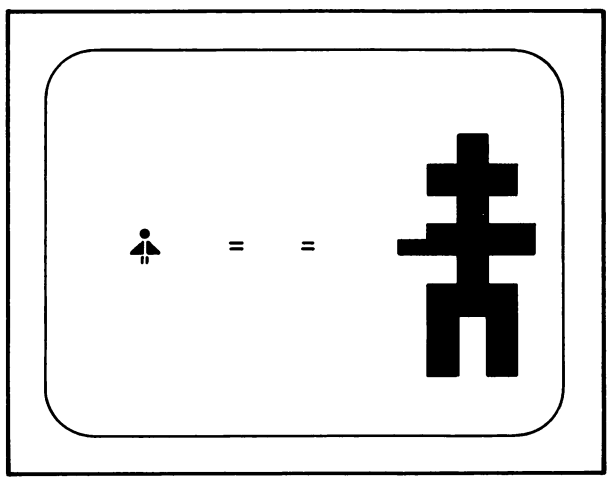
and line 170 to

```
170 CALL VCHAR(X,16,33)
```

Let's put together what we have learned and see if we can use CALL KEY and the IF/THEN statements to shoot at a target.

GUNFIGHTER

This program will allow you to fire a bullet in the direction of a flying bird. You will not be able to hit the bird, but you will be able to see how animation in both directions works. The TI-99/4A computer can only move one character at a time. So, in this program, the bullet will only fire when the bird stops.



We will do more shooting programs in the Arcade chapter. But, the purpose here is to demonstrate animation in both directions. Input the following program.

```
100 CALL CLEAR
110 GOSUB 370
120 GOSUB 200
```

-Sends the program directly to line 370, which plots the cowboy.

-Sends the computer to line 200 to plot the birds.

```
130 FOR X = 26 TO 4 STEP
    -1
140 E = X-1
150 CALL HCHAR(12,X,32)
160 CALL HCHAR(12,E,56)
170 NEXT X
180 GOTO 110
190 REM MOVING BIRD
200 CALL CHAR(40,"10387C
    FE28000000")
210 CALL CHAR(48,"FFFFFFF
    FFFFFFFFF")
220 CALL CHAR(56,"00007
    C80807C0000")
230 CALL SCREEN(5)
240 CALL COLOR(2,2,5)

250 CALL COLOR(4,9,5)

260 FOR B = 24 TO 3 STEP
    -1
270 A = B-1
280 CALL VCHAR(B,5,32)
290 CALL VCHAR(A,5,40)
300 NEXT B
310 FOR C = 1 TO 23
320 D = C+1
330 CALL VCHAR(C,5,33)
340 CALL VCHAR(D,5,40)
350 NEXT C
360 RETURN

370 CALL COLOR(3,15,5)
380 CALL HCHAR(8,27,48)
390 CALL HCHAR(9,26,48,3)
400 CALL HCHAR(10,27,48)
410 CALL HCHAR(11,26,48,3)
420 CALL HCHAR(12,25,48,5)
430 CALL HCHAR(13,27,48)
```

-Lines 130 to 170 display the bullets. See explanation at the end of the program for lines 150 and 160.

-Bird character.

-Character used to build cowboy.

-Character for bullet.

-Colors the screen dark blue.

-Calls all the characters in Set Number 2 black on a dark blue background.

-Colors all the characters in Set Number 4 medium red on a black background.

-Lines 260 to 350 display the bird. See explanation for lines 280, 290, 330, and 340 in the text following the program.

-Sends the computer back to line 120.

-Lines 370 through 450 display the cowboy.

```
440 CALL VCHAR(14,26,48,2)
450 CALL VCHAR(14,28,48,2)
460 RETURN                    -Sends the computer back to line
                               130.
```

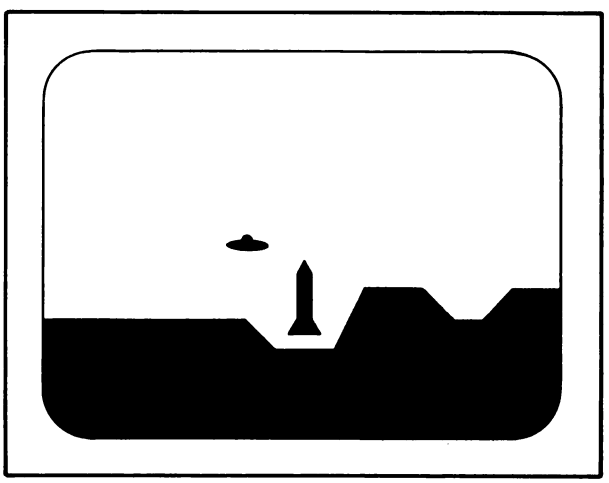
In all of our animation programs thus far, we have used the CALL CLEAR line to erase the character before displaying the next character. We can't use CALL CLEAR after each display in this program because we will erase everything on the screen. So we have to fool both the computer and our eyes. We tell the computer to display the SPACE character, or Number 32 and our desired character alternately, which gives the illusion of movement. By displaying the blank space, we are, in effect, erasing the bird character and the bullet character. Let's look at the bird character. Line 260 tells the computer to begin at row 24, or at the bottom of the screen, and then move up to row 3, in steps of 1.

Line 270 tells the computer to let A = B − 1. Line 280 displays the SPACE. When you are plotting, the SPACE must be displayed first. Line 290 plots the bird character.

Lines 310 to 350 do the same thing for the bird's trip back down the screen. Lines 130 to 170 display the bullets in the same way. This is a good gimmick to remember if you plan to do complicated animation programs.

MARS LANDING

Now let's look at an animated program that we call *Mars Landing.* Try this one.



```
100 CALL CLEAR
110 CALL SCREEN(5)
120 CALL CHAR(33,"FFFFFFF
    FFFFFFFFF")
130 CALL CHAR(34,"80C0E0
    F0F8FCFEFF")
140 CALL CHAR(35,"0103070
    F1F3F7FFF")
150 CALL CHAR(42,"1038383
    87CFE3800")
160 CALL CHAR(72,"00187EF
    F7E3C0000")
170 CALL COLOR(1,9,5)
180 CALL HCHAR(21,1,33,
    128)
190 CALL HCHAR(20,1,33,11)
200 CALL HCHAR(19,1,33,10)
210 CALL HCHAR(18,1,33,9)
220 CALL HCHAR(17,1,33,8)
230 CALL HCHAR(17,9,34)
240 CALL HCHAR(18,10,34)
250 CALL HCHAR(19,11,34)
260 CALL HCHAR(20,12,34)
270 CALL HCHAR(20,17,33,
    15)
280 CALL HCHAR(19,18,33,
    14)
290 CALL HCHAR(18,19,33,
    13)
300 CALL HCHAR(17,20,33,3)
310 CALL HCHAR(20,16,35)
320 CALL HCHAR(19,17,35)
330 CALL HCHAR(18,18,35)
340 CALL HCHAR(17,19,35)
350 CALL HCHAR(17,23,34)
360 CALL HCHAR(17,26,35)
370 CALL HCHAR(17,27,33,6)
380 CALL SOUND(4000,-6,4)
390 FOR X = 1 TO 18
400 A = X + 1
```

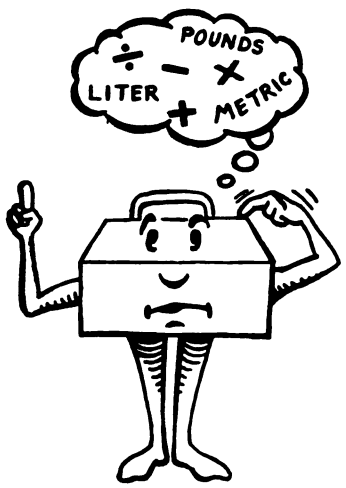
-Lines 120 through 160 define the characters used.

-Lines 180 through 380 display the surface of Mars.

-Lines 390 to 480 display rockets landing.

```
410 CALL VCHAR(A,25,42)
420 CALL VCHAR(X,25,32)
430 NEXT X
440 FOR X = 1 TO 21
450 A = X + 1
460 CALL HCHAR(A,14,42)
470 CALL HCHAR(X,14,32)
480 NEXT X
490 CALL SOUND(2500,-5,0)   -Calls the saucer noise.
500 FOR Y = 21 TO 14 STEP   -Lines 500 through 580 display
    -1                       the saucers.
510 B = Y - 1
520 CALL VCHAR(Y,12,32)
530 CALL VCHAR(B,12,72)
540 NEXT Y
550 FOR Y = 12 TO 2 STEP
    -1
560 A = Y - 1
570 CALL HCHAR(12,6,32)
580 CALL HCHAR(12,A,72)
590 CALL HCHAR(2,12,32)     -Lines 590 and 600 display a blank
600 CALL HCHAR(13,12,32)     where the saucers linger.
610 NEXT Y
620 GOTO 180                -Loops back to line 180.
```

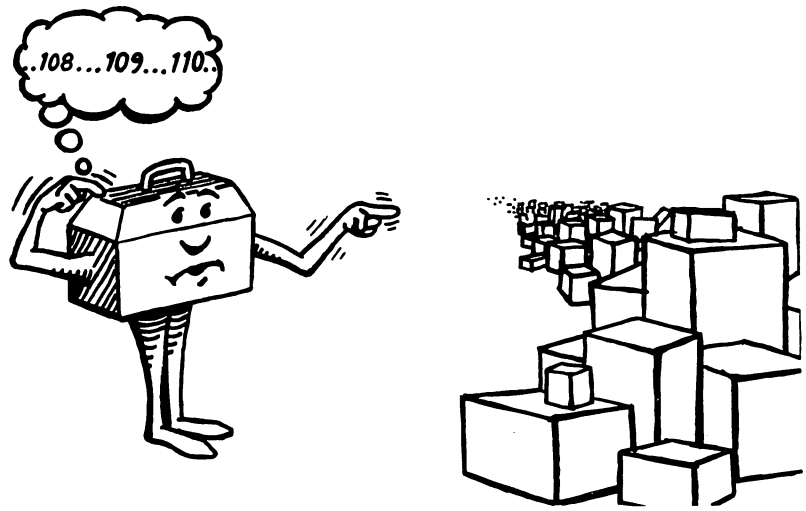
We used the same technique to erase unwanted characters in this program that we used in the Cowboy program. Change this program and experiment. Run the saucers all the way up, or let the rockets take off instead of land. Or you might put some spacemen on the planet's surface. How about having a nasty alien flying around? Your TI-99/4A computer has plenty of memory, so let your imagination go wild.



Calculating Subroutines 7

Your TI-99/4A can easily become a helpful calculator and problem solver. We think it is smart to use your computer in this way. You'll be learning how to develop subroutines for fun-type math games and quizzes in the next chapter.

Don't be anxious about math, because our gentle approach will show you how easy arithmetic and algebra are when worked on the TI computer. Let's start by reviewing simple arithmetic operations.



SIMPLE CALCULATIONS ON THE TI-99/4A

All you have to do to perform simple calculations on your computer is use the PRINT statement. Try it.

PRINT 2 + 2	-(and press ENTER)
PRINT 5 − 2	-(and press ENTER)
PRINT 5 * 2	-(The asterisk (*) is the multiplication sign.)
PRINT 6 / 2	-(The slash is the division sign.)
PRINT 4 ∧ 2	-(The ∧ is used for exponents. 4 ∧ 2 is the same as 4^2.)
PRINT 10 * (4 / 2)	-(The parentheses show the computer which operation to perform first.)

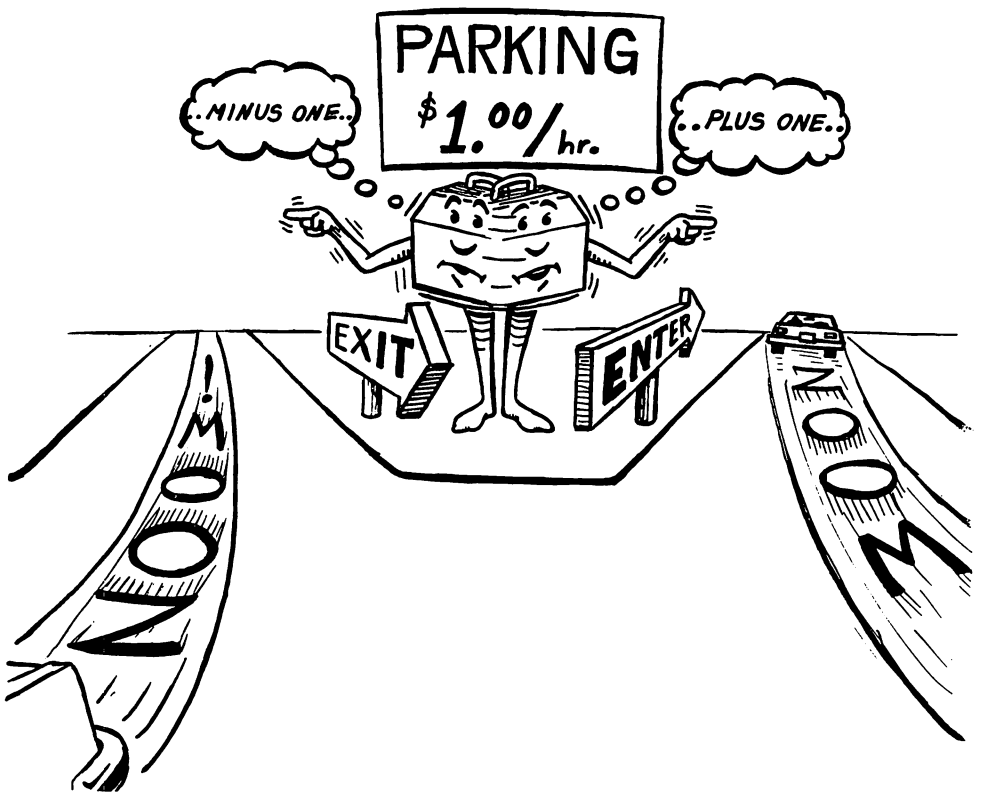
NOTE: Without the parentheses, the computer will follow the rules of priority. These priorities are:

exponent	^	highest priority
multiplication	*	
division	/	
addition	+	
subtraction	−	lowest priority

Let's look at some examples to see how you and the TI-99/4A can handle them.

Parking Lot

At your favorite parking lot, there are 20 rows of cars with 15 cars in each row. How many cars are there?



```
PRINT 20 * 15
300
```

There are 300 cars.

More Problems

In the same full parking lot, four cars leave every hour. Two new cars arrive at the same rate, every hour. After three hours, how many cars are in the lot?

Step-by-step, we can work out the problem:

300 cars in the full lot	300
3 hours times 4 cars leaving	− (3 * 4)
3 hours times 2 cars arriving	+ (3 * 2)

Now, input this line:

```
PRINT 300 - (3 * 4) + (3 * 2)
294
```

There are 294 cars in the lot after three hours.

*Problem 1**

What if 10 cars leave every hour, and 6 arrive at the same rate? How many would there be in 4 hours?

There are other numeric functions your TI-99/4A can perform. We encourage you to learn about square roots, logarithms, cosines, and the like, from the manual that came with your TI-99/4A computer.

CALCULATING SUBROUTINES

We're going to show you how to solve problems on the TI-99/4A by using short programs, or subroutines. First, let's look at the following problem.

Gallons and Liters

If 0.2642 gallon is the same as one liter, how many gallons are in 60 liters?

---

*NOTE: The answers to all the problems are located at the end of the chapter.

We can solve this problem by multiplying 60 liters times 0.2642 to get the correct number gallons. Using the PRINT statement, this would be:

```
PRINT 60 * .2642
```

If we wanted to convert lots of different amounts of liters into gallons, we could write a program (L = liters and G = gallons).

```
5  CALL CLEAR
10 INPUT L                  -Asks you to input a value for L.
20 G = .2642 * L            -Defines G as a function of, or in
                             relation to, L.
30 PRINT G                  -Tells the computers to print G's
                             value.
```

Type this program in and see what happens. The ? (question mark) prompts you to key in an amount of liters and then prints the amount of gallons.

Let's make the program more "friendly." Change line 10 to read:

```
10 INPUT "NUMBER OF LITERS:" : L
```

and change line 30 to read:

```
30 PRINT G; "GALLONS"
```

Now the program proudly asks you to input the number of liters and it displays for you, in plain English, the liters to gallons conversion.

Kilometers and Miles

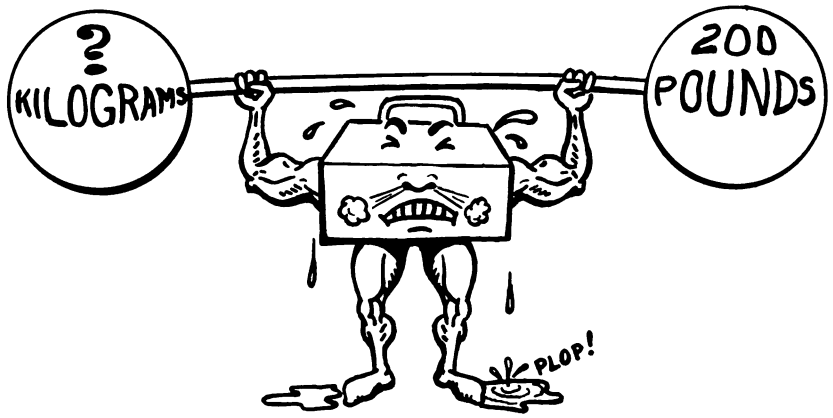
Problem 2

Write a subroutine for converting miles to kilometers. For every kilometer, there is 0.621 mile.

Pounds, Ounces, and Kilograms

Let's do a slightly more complex subroutine by adding another variable to be input.

1 pound = 2.205 kilograms



We want the user to input a number of pounds and ounces and have them converted to kilograms. Try the following program.

```
5  CALL CLEAR                    -Clears screen.
10 PRINT "FIRST POUNDS,          -Instructs you to input pounds
   THEN OUNCES"                    first and then ounces.
20 INPUT P                       -Asks you to give a value to P.
25 INPUT OZ                      -Asks you to give the value of
                                    OZ.
30 PZ = P + OZ / 16              -Defines PZ in relation to pounds
                                    and ounces (P and OZ).
40 KG = PZ / 2.205               -Defines KG in relation to
                                    pounds.
50 PRINT PZ; "POUNDS             -Tells the computer to print
   ARE"; KG; "KILOGRAMS"            pounds and kilograms for the
                                    values you have assigned.
```

Inches and Centimeters

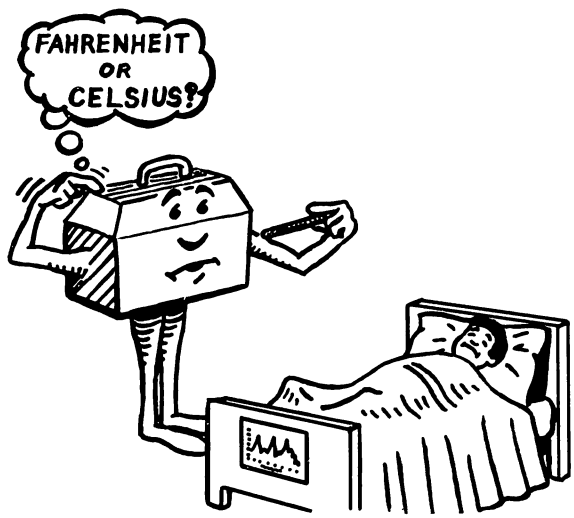
We know that 1 inch equals 2.54 centimeters. Let's write a simple subroutine to solve centimeters.

```
5  CALL CLEAR
10 INPUT "HOW MANY INCHES" : I
20 CM = 2.54 * I
30 PRINT I; "INCHES IS APPROXIMATELY"; CM; "CENTIMETERS"
```

Problem 3

Now..., using what you learned in the pounds/kilograms subroutine, can you change this into a feet, inches, and centimeters conversion subroutine?

Temperature



Converting from Fahrenheit temperatures to Celsius temperatures is not easy. This subroutine should help:

```
5  CALL CLEAR
10 INPUT "DEGREES FAHRE
   NHEIT": F
20 C = 0.55556 * (F-32)
30 PRINT F; "DEGREES FAHR
   ENHEIT IS"; C; "DEGREES
   CELSIUS"
```

-Lines 10 to 30 are similar to other subroutines developed.

```
40 GOTO 10
```
-Tells the computer to go back to line 10 and prompt you for another Fahrenheit value. This line turns the program into a continuous loop; the way to stop it is to press FUNCTION and NUMBER 4 at the same time.

Dollars and Deutschmarks

Have you ever looked at the foreign rates of exchange in the financial section of the newspaper? This subroutine will show you a dollar amount and its equivalent in Marks. In 1983, one dollar was equal to approximately 2.55 Marks in West Germany.

```
10 CALL CLEAR
```
-Starts the program on a clear screen.

```
20 PRINT "DOLLARS", "MARKS"
```
-Tells the computer to divide the screen in half (because of the comma) and put dollars in one side and Marks on the other.

```
30 D = 0
```
-Assigns 0 to D.

```
40 PRINT D, 2.55 * D
```
-Tells the computer to divide the screen in half (because of the comma) and put the value of D on the left and 1.55 * D on the right.

```
50 D = D + 5
```
-Tells the computer to add 5 to its value.

```
60 IF D < 30 THEN 40
```
-Says that as long as D is less than 30, then print both D and 2.55 times D.

```
70 STOP
```
-Tells the program to stop.

Type the program into the computer and RUN it. Do you like the table format?

What happens when you change line 60 to read IF D < 50 THEN 40? How about changing the increment in line 50 from D + 5 to D + 2? Experiment with this table format.

Problem 4

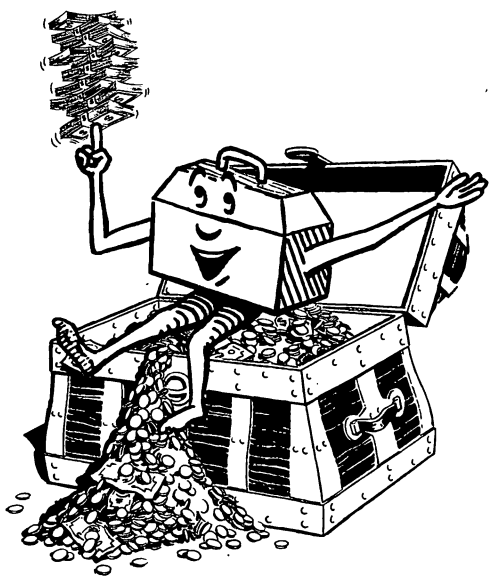
You realize that exchange rates change almost daily. Change the Dollars and Deutschmarks subroutine so that you can input the most current exchange rate.

Problem 5

Wouldn't it be interesting to display Fahrenheit and Celsius temperature conversions in a table format? Go back to the Temperature subroutine and prepare a new program.

Investing

How much money can you make by investing money? This program shows you how easy it is to make money when you have some.



The standard formula for a monetary return is:

$$R = M * (1 + I/100)^Y$$

where,

R = The return,
M = Money invested,

I = Interest rate per period,
Y = The number of periods.

This is a fun subroutine to play with.

```
10 CALL CLEAR
20 PRINT "AMOUNT OF MONEY TO INVEST"
25 INPUT M
30 PRINT "WHAT IS THE INTEREST RATE     PER YEAR?"
35 INPUT I
40 PRINT "HOW MANY YEARS?"
45 INPUT Y
50 R = M * (1 + I/100)^Y
60 PRINT "YOUR RETURN ON"; M; "INVESTED IS"; R;
```

None of the lines used in this subroutine are new. We suggest you add another line so you don't have to type in "NEW" after each run.

```
70 GOTO 20
```

These continuous loops are easy to make. What happens if you had typed in:

```
70 GOTO 10
```

Profit

Most people are in business to make money and making money means profits. How do we figure profit?

Profit = Total Revenues − Total Costs

Let's do a "profitable" subroutine.

```
10 CALL CLEAR
20 PRINT "WHAT WAS YOUR TOTAL REVENUE?"
25 INPUT R
30 PRINT "WHAT WERE YOUR TOTAL COSTS?"
35 INPUT C
40 P = R - C
50 PRINT P; "IS YOUR TOTAL PROFIT"
```

Easy enough. Now, we should figure at what "rate" we are profitable. When comparing businesses, we look at the profit as a percentage

of revenue. By using this ratio, the performances of businesses can be compared regardless of their size. We have to add the following lines to get this ratio:

```
60 PP = P/R * 100
```
–Defines PP (Profit as a percentage of revenue) as profit divided by revenue times 100.

```
70 PRINT PP; "IS YOUR PROF
   IT AS A PERCENTAGE OF
   REVENUE"
```
–Simply prints the value of PP with a message.

Now, RUN the whole program.

Current Ratio

Banks and financial analysts often look at your current ratio as a measure of your "financial health." It boils down to how much you have and what you owe. The simple formula is:

Current Ratio = Current Assets/Current Liabilities

Problem 6

Write a subroutine for figuring current ratio.

Loan Analysis

Using what we know about the current ratio given in the last subroutine, let's figure a way to gauge a person's (or business') credit worthiness in a loan situation.

Some banks won't loan you money if your liabilities exceed your assets (or your current ratio is less than 1). Let's assume that banks will only grant loans if the current ratio is greater than 1.

```
10 CALL CLEAR
20 PRINT "WHAT ARE
   YOUR CURRENT
   ASSETS?"
25 INPUT A
30 PRINT "WHAT ARE
   YOUR CURRENT
   LIABILITIES?"
35 INPUT L
40 R = A/L
```

```
50  IF R > 1 THEN 100
```
-Tells the computer that if the current ratio (R) is greater than 1, it is to go to line 100.

```
60  IF R < = 1 THEN 200
```
-Tells the computer that if R is less than or equal to 1, it is to go to line 200.

```
100 PRINT "YOUR CURRENT
    RATIO IS", R; "YOUR
    LOAN IS APPROVED"
105 STOP
```
-Prints the current ratio and tells you that the loan is accepted.

```
200 PRINT "YOUR CURRENT
    RATIO IS", R; "YOUR
    LOAN IS DENIED"
205 STOP
```
-Prints the current ratio and tells you that the loan is denied.

Problem 7

Using what you have learned in this Loan Analysis case, go back to the Profit subroutine and add lines to the program so that:

-if the percentage of profit is more than 20%, "the company is in good health."
-if the percentage of profit is less than 20%, "the company is performing poorly."

ANSWERS TO PROBLEMS

Problem 1

Parking Lot

```
PRINT 300 - (4 * 10) + (4 * 6)
284
```

Problem 2

Kilometers and Miles

```
 5 CALL CLEAR
10 INPUT "MILES:": M
20 KM = 1.609 * M
30 PRINT M; "MILES EQUAL", KM; "KILOMETERS"
```

Problem 3

Feet, Inches, and Centimeters

```
5  CALL CLEAR
10 PRINT "FIRST FEET, THEN INCHES"
20 INPUT F
25 INPUT I
30 FI = I + (F * 12)
40 CM = 2.54 * FI
50 PRINT FI; "INCHES IS APPROXIMATELY", CM; "CENTIMETERS"
```

Problem 4

Exchange Rate

```
5  CALL CLEAR
10 INPUT "EXCHANGE RATE:": EX
20 PRINT "DOLLARS", "MARKS"
30 D = 0
40 PRINT D, EX * D
50 D = D + 5
60 IF D < 30 THEN 40
70 STOP
```

Problem 5

Temperature Table

```
10 CALL CLEAR
20 PRINT "FAHRENHEIT", "CELSIUS"
30 F = 0
40 PRINT F, 0.55556 * (F-32)
50 F = F + 20
60 IF F < = 220 THEN 40
70 STOP
```

Problem 6

Current Ratio

```
10 CALL CLEAR
20 PRINT "WHAT ARE YOUR CURRENT ASSETS"
25 INPUT A
30 PRINT "WHAT ARE YOUR CURRENT LIABILITIES"
```

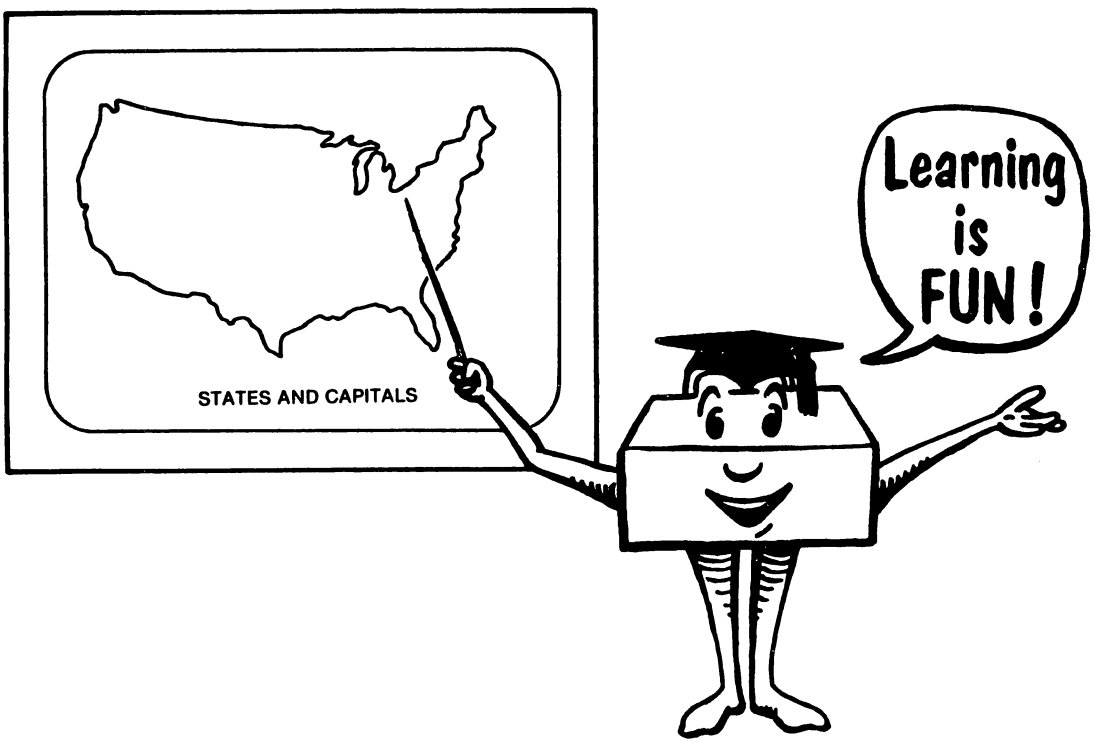
```
35 INPUT L
40 R = A/L
50 PRINT R; "IS YOUR CURRENT RATIO"
```

Problem 7

Profit Analysis

```
70  IF PP < = 20 THEN 100
80  IF PP > 20 THEN 200
100 PRINT "YOUR PERCENTAGE PROFIT IS", PP; "AND YOUR
    COMPANY IS IN GOOD HEALTH"
105 STOP
200 PRINT "YOUR PERCENTAGE PROFIT IS", PP; "AND THE
    COMPANY IS PERFORMING POORLY"
205 STOP
```

We have now covered all of the basic tools. Armed with the knowledge of how the computer works and a variety of subroutines, you are now ready to tackle some real programming. The next three chapters contain thirteen longer programs. Let's begin by creating some educational games.

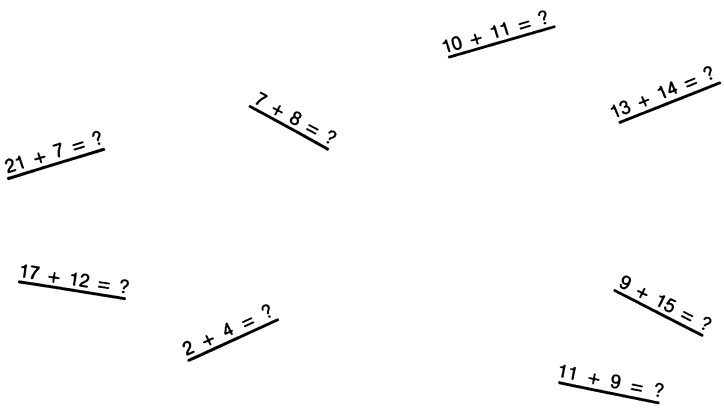


Educational Game Programs 8

The educational games in this chapter are designed so that you can have fun while learning. We will explain the structure of each program and you are encouraged to improve and enhance it with graphics, color, and sound, wherever you like. The logic for these games can be used to create many more games, and we will give you lots of those ideas, too.

One of the main reasons we use a computer and write our own programs is to teach our children math and general educational concepts. Our kids enjoy quizzing each other and their parents about all kinds of things, from states and capitals to riddles. We hope you have as good a time with these games as we did.

ARITHMETIC TESTER



This program asks you to add numbers. It stops the game after 10 questions, gives you the score, and ranks you on three levels of competence. Type the program in and try it out.

```
100 CALL CLEAR
110 CALL SCREEN(5)
120 FOR SET = 1 TO 8
130 CALL COLOR(SET,16,5)
140 NEXT SET
150 PRINT "ADD THESE NU
    MBERS"
160 FOR T = 1 TO 1000
170 NEXT T
```

-Clears the screen.
-Colors the screen dark blue.
-Lines 120 to 140 print all characters in sets 1 to 8 as white on a dark blue background.
-Lines 150 to 170 are the first prompt.

```
180 Q = 0
190 N = 0
200 RANDOMIZE
210 A = INT(10 * RND)
220 B = INT(10 * RND)
230 IF Q = 10 THEN 420
240 Q = Q + 1
250 CALL CLEAR
260 PRINT "WHAT IS THE
    SOLUTION FOR";A;"PL
    US";B
270 PRINT
280 INPUT X
290 PRINT
300 IF X = A + B THEN 360
310 PRINT "WRONG"
320 PRINT "(A+B); IS THE
    CORRECT ANSWER"
330 FOR T = 1 TO 500
340 NEXT T
350 GOTO 400
360 PRINT "CORRECT"
370 FOR T = 1 TO 500
380 NEXT T
390 N = N + 1
400 PRINT
410 GOTO 210
420 GOTO 430
430 PRINT N; "QUESTIONS
    RIGHT OUT OF"; Q
440 J = N/Q
450 IF J = 1 THEN 490
460 IF (J < 1) * (J > =.7)
    THEN 530
```

-Lines 180 and 190 are the initial values of Q and N.

-Tells the TI computer to randomize.

-Lines 210 and 220 define the values of A and B as a random number from 0 to 10.

-If 10 questions are answered, the program goes to 420.

-Keeps counting the number of questions.

-Clears the screen.

-Screen prompt.

-Allows you to input the solution.

-If you answer correctly, the computer goes to 360.

-If you are wrong, this prints "Wrong".

-Gives the correct answer.

-"Correct" screen prompt.

-Loops back and gets two more numbers to add.

-Prints your score.

-Computes score.

-Branches to 490 if J = 1.

-Branches to 530 if you miss 3 or less.

```
470 IF J < .7 THEN 570          -Branches to 570 if you miss 3 or
480 END                           more.
490 PRINT "VERY GOOD            -Perfect score prompt.
    WORK"
500 FOR T = 1 TO 1000           -Lines 500 and 510 hold the
510 NEXT T                        prompt on the screen.
520 GOTO 480
530 PRINT "GOOD WORK,           -Prompt for missing 3 or less.
    BUT MORE PRACTICE
    WILL HELP"
540 FOR T = 1 TO 1000
550 NEXT T
560 GOTO 480
570 PRINT "YOU NEED TO          -Prompt for missing 3 or more.
    PRACTICE A LOT"
580 FOR T = 1 TO 1000
590 NEXT T
600 GOTO 480
```

SUGGESTED PROGRAM CHANGES AND "DRESSING"

You can enhance or "dress up" the program by adding sound, color, and graphics. Look at your "tools" chapters for ideas. In this program, there are several natural places for program enhancement:

1. When a correct answer is entered.
2. When an incorrect answer is entered.
3. When the score for the round is displayed using lines 490, 530, and 570.

The enhancement can be as simple as a different screen and graphic combination or as complex as a smiling face with changing background colors and music for a job well done.

The program can be easily altered to suit your needs in many ways.

1. **LEVEL**—Change the level of difficulty by changing lines 210 and 220. To make the quiz harder, change the statement to read:

```
A = INT(20 * RND)
B = INT(20 * RND)
```

This will give you addition problems with numbers from 0 to 20.

2. **LENGTH**—You can make the quiz longer by changing the number of questions asked before the program ends. Change line 230 to:

```
230 IF Q = 20 THEN 420
```

   This will give you a quiz with 20 questions. You can change the number 20 to any number to get a quiz of any length.

3. **GRADING**—You can set your own grades or competency levels by changing lines 450 through 470. If you would rather have "VERY GOOD WORK" mean 0.9 (or 90%) and higher, change the lines to read:

```
450 IF J >= 0.9 THEN 490
460 IF (J < 0.9) * (J > 0.7) THEN 520
470 IF J < 0.7 THEN 570
```

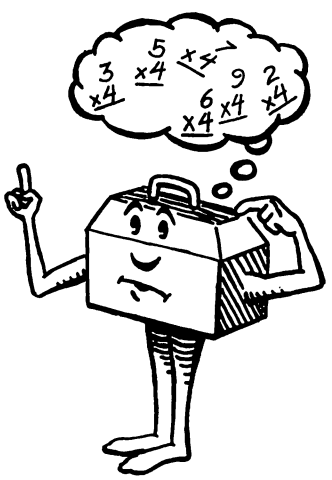
4. **OPERATION**—You can change this game from addition to any other arithmetic operation that you want. For subtraction, change line 300 to:

```
IF X = A - B THEN 360
```

   and line 320 to:

```
PRINT (A-B); IS THE CORRECT ANSWER"
```

   You will also need to change the screen prompts to read minus rather than plus.



MULTIPLICATION TABLES

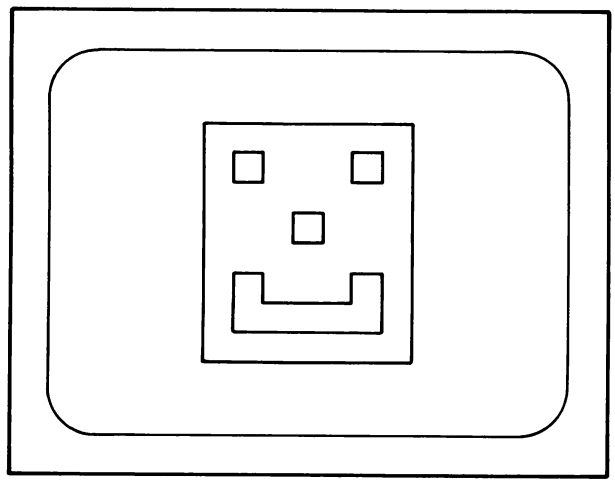
Using the same program structure as shown in the "Arithmetic Tester" program given at the beginning of the chapter, along with READ and DATA statements, you can quiz your knowlege of the multiplication tables. The number of questions asked is the same as the amount of data. When your answers are all correct, the computer smiles for you.

```
100 CALL CLEAR                    -Clears the screen.
110 PRINT "THIS PROGRAM           -Lines 110 to 170 are the begin-
    WILL HELP YOU                  ning screen prompt.
    LEARN"
120 PRINT
130 PRINT "YOUR MULTIPLI
    CATION TABLES"
140 PRINT
150 "HAVE FUN"
160 FOR T = 1 TO 800
170 NEXT T
180 Q = 0                         -Initial value of Q.
190 N = 0                         -Initial value of N.
200 READ A                        -Tells the computer to read the
                                   first statement in the data state-
                                   ment.
210 IF A = 99 THEN 410            -If 99 is the last number in the
                                   data statement, the program will
                                   branch to 410.
220 READ B                        -Reads the second number.
230 Q = Q + 1                     -Calculates the number of ques-
240 CALL CLEAR                     tions.
250 PRINT "WHAT IS THE A          -Displays the question to be
    NSWER FOR";A;"TIMES;B;         solved.
260 PRINT
270 INPUT X                       -Allows you to input the answer.
280 PRINT
290 IF X = A * B THEN 350         -Correct answer condition.
300 PRINT "WRONG"                 -Lines 300 to 330 are the wrong
310 PRINT (A * B); "IS THE         answer prompt.
    CORRECT ANSWER"
320 FOR T = 1 TO 500
```

```
330 NEXT T
340 GOTO 390
350 PRINT "CORRECT"          -Correct prompt.
360 FOR T = 1 TO 500         -Holds the correct prompt on the
370 NEXT T                    screen.
380 N = N + 1
390 PRINT
400 GOTO 200
410 GOTO 420
420 PRINT N; "QUESTIONS      -Displays score.
    RIGHT OUT OF"; Q
430 J = N/Q                  -Calculates the score.
440 IF J = 1 THEN 490        -Lines 440 to 460 branch to the
450 IF (J < 1) * (J > =.6) TH appropriate prompt.
    EN 640
460 IF J < .6 THEN 680
470 DATA 11,3,12,6,15,9,14,  -Data.
    4,10,11,13,12,15,16,11,
    12,13,5,7,14,99
480 END
490 PRINT "WAY TO GO"        -100% condition.
500 CALL CHAR(96,"FFFFFFF    -Lines 500 to 600 display the smil-
    FFFFFFFF")                ing face.
510 CALL VCHAR(7,11,96,12)
520 CALL HCHAR(18,12,96,
    10)
530 CALL VCHAR(7,21,96,11)
540 CALL HCHAR(7,12,96,9)
550 CALL HCHAR(9,14,96)
560 CALL HCHAR(9,18,96)
570 CALL HCHAR(12,16,96)
580 CALL HCHAR(15,14,96,5)
590 CALL HCHAR(14,14,96)
600 CALL HCHAR(14,18,96)
610 FOR I = 1 TO 1000        -Holds smiling face on the screen.
620 NEXT I
630 GOTO 480
640 PRINT "GOOD WORK,        -Prompt for score of 60% or bet-
    PRACTICE MAKES PER        ter.
    FECT"
```

```
650 FOR I = 1 TO 500
660 NEXT I
670 GOTO 480
680 PRINT "NOT SO HOT,
    TRY AGAIN"
690 FOR I = 1 TO 500
700 NEXT I
710 GOTO 480
```

-Prompt for score of less than 60%



Suggested Program Changes

You can add sound, color, and more graphics to the scoring prompts. We have used a smiling face to encourage you to be creative. Try adding a frowning face after line 680. Use the Random Color subroutine from Chapter 3 to reinforce a good score. Or, add a little sound.

You can change the program to suit yourself. Increase or decrease the level of difficulty of the problems by changing the numbers in the DATA statement. Or, add more numbers in the DATA statement to make the quiz last as long as you like. Or, try changing the scoring. Maybe you can develop your own point system. Change the operation to use addition, division, and subtraction problems.

The READ and DATA statements are pretty powerful statements and we will use them again with our States and Capitals quiz that is given later on.

RIDDLE FRACTIONS

This riddle game is fun and it tests your ability to understand fractions. The computer asks you to figure out clues to the riddle by using your knowledge of fractions. Then, you have to use your imagination to answer the riddle! For example:

MIDDLE 1/2 OF MATE = AT
FIRST 1/2 OF RUNNER = RUN

If you cannot answer the riddle, key in anything and the program will give you a hint. You must answer the question correctly with the hint. The computer will keep asking you *forever,* using the hint, until you finally get the riddle—just like your friends do when they tell you a riddle! Try it!

```
100 CALL CLEAR                    -Clears the screen.
110 READ A$                       -Tells the TI computer to read A$
                                   from DATA.
120 IF A$ = "END" THEN 350        -When all DATA is read stop.
130 READ B$,C$,D$,E$,F$,          -Reads from DATA.
    G$,H$,I$,J$,K$
140 READ L$,M$,N$
150 PRINT"SOLVE THIS RI           -Beginning screen prompt.
    DDLE"
160 PRINT
170 PRINT A$;B$;C$;D$;E$;         -Prints clues.
    F$;G$;H$;I$;J$;
180 INPUT X$                      -Allows you to type in answer to
                                   the riddle.
190 IF X$ = K$ THEN 260           -Correct guess condition.
200 PRINT "HERE IS A HINT,        -Hint prompt.
    TRY AGAIN"
210 PRINT
220 PRINT L$;M$;N$                -Prints the hints.
230 INPUT Y$                      -Allows for another guess.
240 IF Y$ = K$ THEN 260           -Correct second answer condi-
                                   tion.
250 GOTO 200                      -Loops back to 200 if second guess
                                   isn't correct.
```

```
260 PRINT "YOU ARE RIGHT.          -Correct guess prompt.
    THE ANSWER WAS"; K$
270 FOR T = 1 TO 1000
280 NEXT T
290 GOTO 250
300 DATA FIRST 1/2 OF WH           -Lines 300 to 340 are the DATA
    ATEVER, MIDDLE 3/5 OF           from which the clues, hints, and
    CHASE, FIRST 1/2 OF             the answer are printed.
    FIVEFOLD
310 DATA LAST 1/2 OF BUC
    KEYES, MIDDLE 3/5 OF
    HANDY, LAST 5/7 OF
    ARRESTS
320 DATA LAST 1/3 OF BE
    ACON, MIDDLE 1/9 OF
    CRABAPPLE, FIRST 5/9
    OF WATERFALL
330 DATA LAST 3/7 OF GRA
    BBED, THE MISSISSIPPI
    RIVER, MIDDLE 3/5 OF
    OTHER
340 DATA FIRST 11/13 OF
    MISSISSIPPIAN, LAST 5/6
    OF DRIVER
350 END                            -Ends the program.
```

Add color, sound and more elaborate graphics if you like.

This program is persistent about asking for the answer. It won't stop trying. If you would rather have the answer to the riddle appear after you fail to answer correctly using the hint, here is how to do it:

```
250 PRINT "WRONG AGAIN. THE ANSWER IS"; K$
255 GOTO 350
```

We recommend that you use your own riddles in the DATA statements as turning the riddles into the "fraction code" is half the fun. The other half is trying them out on your friends. The following riddles are corny, but maybe they will bring to mind some of your favorites.

What kind of bridge makes people most anxious?
ANSWER: A suspension bridge.

What has teeth but cannot bite or chew?
ANSWER: A comb.

What does a pet canary say on Halloween?
ANSWER: Twick or Tweet.

What do you get when you cross a centipede with a parrot?
ANSWER: A walkie-talkie.

Remember that in DATA statements, you can only fill four lines before you must add another line number.

COLORED PENS

The following program received more attention from the kids in our neighborhood than any other in the book. It is called COLORED PENS. There is no score to keep, just plain artistic fun. By using different keys, you can draw in four different colors. The color and direction keys are listed in the next section so that you can get familiar with them.

Color and Direction Codes

1. RED
   W = Up
   A = Left
   S = Right
   Z = Down
2. BLUE
   R = Up
   D = Left
   F = Right
   V = Down
3. GREEN
   Y = Up
   G = Left
   H = Right
   B = Down
4. YELLOW
   I = Up

J = Left
K = Right
M = Down

5. ERASE
P = Up
L = Left
; = Right
. = Down

The Erase function will take away anything you don't like, one square at a time. The space bar will clear the entire screen.

A Word About the Program

Look at lines 4030 through 4060. These lines are the backbone of the program. Line 4030 tells the computer that if the W key is not pressed, to go on to line 4070 and check for the Z key, and so on, through all the possible keys. If W is pressed, or if A = W, then lines 4040 through 4060 will be executed.

The computer will accept a key and will then run through all the lines, like 4030, until it finds the right key and, then, will execute those lines, like 4040 through 4060. These lines plot the PEN in the correct direction and in the proper color.

Now let's input it and have some fun.

```
100  CALL CLEAR
110  GOSUB 12000
120  C = 16
130  R = 22
4000 OLDR = R                    -Defines Old R as equal to R so
                                  that the drawing will continue.
4010 OLDC = C                    -Defines Old C as equal to C for
                                  the same reason.
4020 CALL KEY(0,A,STATUS)        -Activates the keyboard so we
                                  can direct the pen.
4030 IF A<>ASC("W")              -RED up.
     THEN 4070
4040 CHAR = 96
4050 CALL COLOR(9,7,7)
4060 R = R-SGN(R-1)
4070 IF A<>ASC("Z")              -RED down.
     THEN 5010
```

```
4080 CHAR = 96
4090 CALL COLOR(9,7,7)
5000 R = R+SGN(22-R)
5010 IF A<>ASC("A")          -RED left.
     THEN 5050
5020 CHAR = 96
5030 CALL COLOR(9,7,7)
5040 C = C-SGN(C-2)
5050 IF A<>ASC("S")          -RED right.
     THEN 5090
5060 CHAR = 96
5070 CALL COLOR(9,7,7)
5080 C = C+SGN(31-C)
5090 IF A<>ASC("R")          -BLUE up.
     THEN 6030
6000 CHAR = 104
6010 CALL COLOR(10,5,5)
6020 R = R-SGN(R-1)
6030 IF A<>ASC("V")          -BLUE down.
     THEN 6070
6040 CHAR = 104
6050 CALL COLOR(10,5,5)
6060 R = R+SGN(22-R)
6070 IF A<>ASC("D")          -BLUE Left.
     THEN 7010
6080 CHAR = 104
6090 CALL COLOR(10,5,5)
7000 C = C-SGN(C-2)
7010 IF A<>ASC("F")          -BLUE right.
     THEN 7050
7020 CHAR = 104
7030 CALL COLOR(10,5,5)
7040 C = C+SGN(31-C)
7050 IF A<>ASC("Y")          -GREEN up.
     THEN 7090
7060 CHAR = 112
7070 CALL COLOR(11,13,13)
7080 R = R-SGN(R-1)
7090 IF A<>ASC("B")          -BLUE down.
     THEN 8030
```

```
8000  CHAR = 112
8010  CALL COLOR(11,13,13)
8020  R = R+SGN(22-R)
8030  IF A<>ASC("G")          -GREEN left.
      THEN 8070
8040  CHAR = 112
8050  CALL COLOR(11,13,13)
8060  C = C-SGN(C-2)
8070  IF A<>ASC("H")          -GREEN right.
      THEN 9010
8080  CHAR = 112
8090  CALL COLOR(11,13,13)
9000  C = C+SGN(31-C)
9010  IF A<>ASC("I")          -YELLOW up.
      THEN 9050
9020  CHAR = 120
9030  CALL COLOR(12,11,
      11)
9040  R = R-SGN(R-1)
9050  IF A<>ASC("M")          -YELLOW down.
      THEN 9090
9060  CHAR = 120
9070  CALL COLOR(12,11,
      11)
9080  R = R+SGN(22-R)
9090  IF A<>ASC("J")          -YELLOW left.
      THEN 10030
10000 CHAR = 120
10010 CALL COLOR(12,11,
      11)
10020 C = C-SGN(C-2)
10030 IF A<>ASC("K")          -YELLOW right.
      THEN 10070
10040 CHAR = 120
10050 CALL COLOR(12,11,
      11)
10060 C =C+SGN(31-C)
10070 IF A<>ASC("P")          -Erase up.
      THEN 10110
```

```
10080 CHAR = 128
10090 CALL COLOR(13,15,
      15)
10100 R = R-SGN(R-1)
10110 IF A<>ASC(".")
      THEN 10150
10120 CHAR = 128
10130 CALL COLOR(13,15,
      15)
10140 R = R+SGN(22-R)
10150 IF A<>ASC("L")
      THEN 10190
10160 CHAR = 128
10170 CALL COLOR(13,15,
      15)
10180 C = C-SGN(C-2)
10190 IF A<>ASC(";")
      THEN 10230
10200 CHAR = 128
10210 CALL COLOR(13,15,
      15)
10220 C = C+SGN(31-C)
10230 IF A<>ASC(" ")
      THEN 10250
10240 CALL CLEAR
10250 IF (R=OLDR) *
      (C=OLDC) THEN
      10270
10260 CALL GCHAR(R,C,A)
10270 CALL VCHAR(R,C,
      CHAR)
10280 GOTO 4000
12000 CALL SCREEN(15)
12010 CALL CHAR(96,"FFFFF
12020 CALL CHAR(104,"FFFF
      FFFFFFFFFFFF")
12030 CALL CHAR(112,"FFFF
      FFFFFFFFFFFF")
12040 CALL CHAR(120,"FFFF
```

-ERASE down.

-ERASE left.

-ERASE right.

-Clears screen.

-Branches plot the square.

-Plots the first square.

-Plots the squares after the first one and creates the pen's stroke.

-Colors the screen white.

-Lines 12000 to 12050 define the same character with different values so we can call different colors using the same character.

```
      FFFFFFFFFFFF")
12050 CALL CHAR(128,"FFFF
      FFFFFFFFFFFF")
12070 PRINT "LEARN TO
      DRAW"
12080 PRINT
12090 PRINT "WITH THE TI
      COMPUTER"
12100 FOR I = 1 TO 1000
12110 NEXT I
12120 CALL CLEAR
12130 PRINT "JUST PRESS
      THE KEYS FOR"
12140 PRINT
12150 PRINT "THE RIGHT
      COLOR AND"
12160 PRINT
12170 PRINT "DIRECTION"
12180 FOR I = 1 TO 1000
12190 NEXT I
12200 CALL CLEAR
12210 PRINT "  (4 spaces) W"
12220 PRINT "A (2 spaces)
      RED (2 spaces) S"
12230 PRINT "  (4 spaces) Z"
12240 PRINT
12250 PRINT "  (5 spaces) R"
12260 PRINT "D (2 spaces)
      BLUE (2 spaces) F"
12270 PRINT "  (5 spaces) V"
12280 PRINT
12290 PRINT "  (5 spaces) Y"
12300 PRINT "G (2 spaces)
      GREEN (2 spaces) H"
12310 PRINT "  (5 spaces) B"
12320 PRINT
12330 PRINT "  (6 spaces) I"
12340 PRINT "J (2 spaces)
      YELLOW (2 spaces) K"
12350 PRINT "  (6 spaces) M"
```

–Lines 12070 to 12550 display the beginning screen prompts.

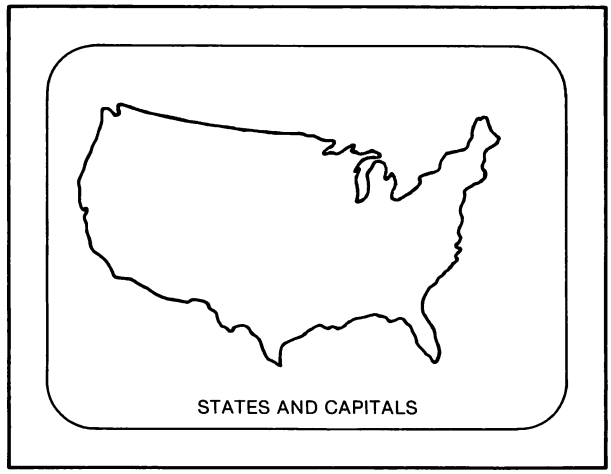
–The number of spaces within parentheses in lines 12210 through 12390 are to be entered with the SPACEBAR. Do not type in "(4 spaces)," "(2 spaces)," etc.

```
12360 PRINT
12370 PRINT " (5 spa-
      ces)       P"
12380 PRINT "L (2 spaces)
      ERASE (2 spaces) ;"
12390 PRINT " (5 spaces) ."
12400 FOR I = 1 TO 2000
12410 NEXT I
12420 PRINT "COPY THE
      KEY CODES ON
      PAPER"
12430 PRINT
12440 PRINT "UNTIL YOU
      GET USED TO THEM."
12450 PRINT
12460 PRINT "THEY ARE IN
      THE BOOK."
12470 FOR I = 1 TO 1000
12480 NEXT I
12490 CALL CLEAR
12500 PRINT "PRESS ANY
      OF THE CODE KEYS"
12510 PRINT
12520 PRINT "TO BEGIN"
12530 FOR I = 1 TO 800
12540 NEXT I
12550 CALL CLEAR
12560 RETURN              -Returns to main body of the
                           program.
```

You might want to experiment with different characters or brushes. Try designing a thinner brush or maybe a diamond-shaped image. Just redefine the character in lines 12010 through 12050.

STATES AND CAPITALS

This is our centerpiece educational game because it provides such a useful program structure for learning the states and capitals as well as for many other educational games.



You are asked to name the capitals of all the 50 states and are then scored and evaluated at the end of the game. Color and sound will both greet you in this game in the scoring subroutines, as well as when you enter an incorrect or correct response.

The structure of the overall program is similar to that used in Arithmetic Tester and Multiplication Tables. It uses a powerful BASIC tool—*arrays*. With the DIM statement in this program, you are allowed to dimension the data in two ways—one for states and one for capitals. Because we use Arrays, DATA statements, and the Randomize Command, we are able to answer questions that are asked randomly.

```
100 CALL CLEAR                  -Clears the screen.
110 FOR I = 1 TO 24             -Lines 110 to 130 plot and scroll
120 PRINT "***** STATES           the beginning screen prompt.
    AND CAPITALS *****"
130 NEXT I
140 GOSUB 310                   -Branches to 310.
150 FOR I = 1 TO 65             -Number of times that DATA is to
                                   be read.
160 READ R,C,CH,N               -Reads from DATA.
170 CALL HCHAR(R,C,CH,N)        -Graphics subprogram that the
180 NEXT I                         DATA will fill.
```

```
190 GOTO 440
200 DATA 5,4,102,1,5,5,101,
    1,6,4,101,1,6,5,96,4,6,
    9,98,2,6,11,100,3,5,30,
    100,1
210 DATA 6,29,101,1,6,30,96,
    1,7,4,101,1,7,5,96,13
220 DATA 7,4,101,1,7,5,96,
    13,7,18,98,2,7,20,96,1,7,
    21,102,1,7,29,101,1,7,30,
    96,1
230 DATA 8,4,101,1,8,5,96,
    17,8,22,98,2,8,27,100,1,8,
    28,98,1,8,29,96,1,8,30,
    102,1
240 DATA 9,3,102,1,9,4,96,
    21,9,25,100,1,9,26,98,1,9,
    27,96,3
250 DATA 10,3,101,1,10,4,96,
    26,10,29,102,1,11,3,96,
    26,12,3,96,26,13,3,96,25
260 DATA 13,28,102,1,14,3,
    101,1,14,4,96,24,14,28,
    102,1,15,3,101,1,15,4,96,
    24
270 DATA 16,3,96,24,16,27,
    102,1,15,5,101,1,17,6,96,
    21,18,6,96,18,19,8,96,18
280 DATA 20,10,101,1,20,11,
    96,15,21,11,101,1,21,12,
    96,1,21,13,97,1,21,14,96,
    4
290 DATA 21,18,97,3,21,25,
    96,1,21,26,102,1,22,14,
    96,3,22,17,102,1,22,25,
    101,1
300 DATA 22,25,96,1,23,14,
    101,1,23,15,96,1,23,16,
    102,1,23,26,97,1
```

-Lines 200 to 300 are the row and column characters and repeat numbers, which plot the map.

```
310 CALL CHAR(96,"0000000
    000000000")
320 CALL CHAR(97,"FFFFFFF
    F00000000")
330 CALL CHAR(98,"0000000
    0FFFFFFFF")
340 CALL CHAR(99,"FF00000
    000000000")
350 CALL CHAR(100,"000000
    00000000FF")
360 CALL CHAR(101,"0F0F0F
    0F0F0F0F0F")
370 CALL CHAR(102,"F0F0F0
    F0F0F0F0F0")
380 CALL SCREEN(5)
390 CALL COLOR(9,7,7)
400 FOR SET = 3 TO 8
410 CALL COLOR(SET,16,5)
420 NEXT SET
430 RETURN
440 CALL CLEAR
450 Q = 0
460 N = 0
470 RANDOMIZE
480 DIM A$(50), B$(50)
490 FOR Z = 1 TO 50
500 READ A$(Z), B$(Z)
510 NEXT Z
520 Z = INT(RND * 50) + 5
530 IF Z = 0 THEN 520
540 IF Q = 20 THEN 630
550 Q = Q + 1
560 PRINT "WHAT IS THE
    CAPITAL OF (1 Space)";
    A$(Z)
570 PRINT
580 INPUT X$
590 PRINT
```

-Lines 310 to 370 define the characters used to plot the map.

-Calls screen blue.

-Colors the map red.

-Lines 400 to 420 color all numbers and letters white.

-Initial value of Q.

-Initial value of N.

-Dimensions the array. Lines 480 to 510 read a state and capital at random.

-Random number generator.

-The 20 is the number of questions in the quiz.

-Allows input.

```
600 IF X$ = B$(Z) THEN 820
610 IF X$ <> B$(Z) THEN
    870
620 GOTO 520
630 GOTO 640
640 PRINT N; "QUESTIONS
    RIGHT OUT OF (1
    Space)"; Q
650 J = N/Q
660 IF J >= 0.9 THEN 900
670 IF (J < 0.9) * (J >= 0.7)
    THEN 960
680 IF J < 0.7 THEN 1020
690 DATA ALABAMA,MON
    TGOMERY,NEW YORK,
    ALBANY,NEVADA,CAR
    SON CITY,OHIO,COL
    UMBUS
700 DATA IOWA,DES MOIN
    ES,SOUTH CAROLINA,
    COLUMBIA,OREGON,
    SALEM,TEXAS,AUSTIN
710 DATA KENTUCKY,FRAN
    KFORT,NEW JERSEY,TR
    ENTON,MASSACHUSE
    TTS,BOSTON,ARIZONA
    ,PHOENIX
720 DATA TENNESSEE,NA
    SHVILLE,WYOMING,CH
    EYENNE,MINNESOTA,
    ST.PAUL,MARYLAND,
    ANNAPOLIS
730 DATA NEW MEXICO,SA
    NTA FE,NORTH CAROL
    INA,RALEIGH,CONNEC
    TICUT,HARTFORD,ILLIN
    OIS,SPRINGFIELD
740 DATA MAINE,AUG
```

-Correct guess condition.
-Incorrect guess condition.

-Lines 640 to 680 display, compute, and branch, depending on score.

```
      USTA,MICHIGAN,LAN
      SING,GEORGIA,ATLA
      NTA,ALASKA,JUNEAU,
      MONTANA,HELENA
750 DATA ARKANSAS,LIT
      TLE ROCK,CALIFORNIA,
      SACRAMENTO,COLOR
      ADO,DENVER,DELAW
      ARE,DOVER
760 DATA FLORIDA,TALLA
      HASSEE,HAWAII,HON
      OLULU,IDAHO,BOISE,
      INDIANA,INDIANAPO
      LIS
770 DATA KANSAS,TOPEKA
      ,LOUISIANA,BATON RO
      UGE,MISSISSIPPI,JACK
      SON,MISSOURI,JEFFER
      SON CITY
780 DATA UTAH,SALT LAKE
      CITY,NEBRASKA,LINC
      OLN,NEW HAMPSHIRE,
      CONCORD,NORTH DA
      KOTA,BISMARCK
790 DATA OKLAHOMA,OK
      LAHOMA CITY,PENN
      SYLVANIA,HARRISB
      URG,RHODE ISLAND,
      PROVIDENCE
800 DATA SOUTH DA
      KOTA,PIERRE,VERMO
      NT,MONTPELIER,VIRG
      INIA,RICHMOND,WAS
      HINGTON,OLYMPIA
810 DATA WEST VIRGINIA,
      CHARLESTON,WISC
      ONSIN,MADISON
820 PRINT "CORRECT"
```

-Lines 820 to 850 are the correct answer prompt.

```
830 CALL SOUND(500,262,
    0)
840 CALL SOUND(500,292,
    0)
850 N = N + 1
860 GOTO 610
870 PRINT "WRONG, THE
    ANSWER IS"; B$(Z)
880 CALL SOUND(500,-1,
    0)
890 GOTO 620
900 PRINT "YOU KNOW
    YOUR STATES"
910 PRINT
920 PRINT "AND CAPI
    TALS"
930 FOR I = 1 TO 500
940 NEXT I
950 END
960 PRINT "OK, BUT YOU
    COULD"
970 PRINT
980 PRINT "USE MORE
    STUDY"
990 FOR I = 1 TO 500
1000 NEXT I
1010 END
1020 PRINT" YOU NEED
    MORE STUDY"
1030 FOR I = 1 TO 500
1040 NEXT I
1050 END
```

-Lines 870 to 890 are the wrong answer prompt.

-Lines 900 to 950 are the 100% to 90% score prompt.

-Lines 960 to 990 are the 89% to 70% score prompt.

-The 69% or below score prompt.

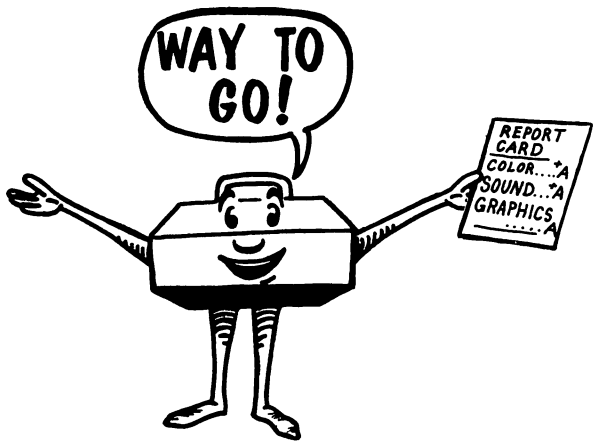
Suggested Changes to the "States and Capitals" Program

We have included color and sound subroutines in this game. You should improve on them for yourself. There are a number of ways to adapt this game.

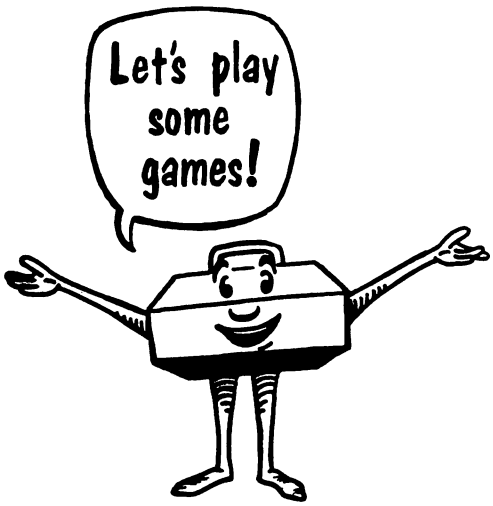
1. Change the DATA for the states and capitals. Either more DATA or less DATA will also change the game.
2. Change the scoring levels.

This is the last educational game that we will use as an illustration. We hope that you now have lots of ideas to use for games and quizzes. Some of the different ideas that we would like to suggest are:

Countries and Capitals
Continents and Countries
Inventors and Objects
Historical Events and Dates
Mythological Characters
Currency and Countries
Biblical Names
Diseases and Their Meanings
Music and Musicians
Chemical Elements and Symbols
Countries and Leaders
Words and Their Opposites
Numbers and Their Spelling
Roman Numerals and Numbers
Works of Art and Artists



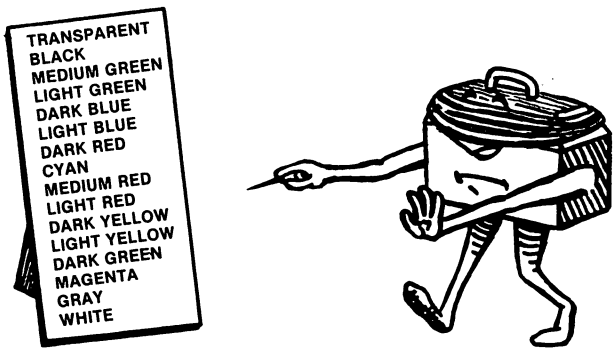
In the next chapter on traditional games, we will expand our knowledge of BASIC programming while continuing to have fun.



Traditional Game Programs 9

For our purposes, Traditional Games are games that have been played for years with dice, cards, paper and pencil, or gameboards. In this chapter, we will look at four such games and will change them so that they can be played with your TI computer. Using these games as models, you should be able to create or adapt other traditional games for the TI-99/4A. Let's begin with a very easy game.

One of the oldest and simplest traditional computer games is the Guessing Game. In the Guessing Game, you can guess colors, sounds, numbers, or just about anything else you want to try. Regardless of what is guessed, the program will be about the same. The computer picks a number, color, or whatever, by using the Randomize command and the Random function. Then, you try and guess the computer's choice by using either Input statements or CALL KEY. The computer checks to see if you guessed correctly or incorrectly and lets you know the outcome.



Let's look at a Color Guessing game.

```
100 CALL CLEAR
110 RANDOMIZE
120 X = INT(16*RND)
130 TRYS = 0
140 IF X = 0 THEN 120
150 INPUT "PICK A COLOR USING THE TI COLOR CODE
    NUMBERS": Y
160 PRINT
170 TRYS = TRYS + 1
180 PRINT "THAT WAS TRY NUMBER     "; TRYS
190 IF X<>Y THEN 280
200 CALL CLEAR
```

```
210 CALL SCREEN(X)
220 PRINT "YOU GUESSED IT CORRECTLY"
230 PRINT "IT ONLY TOOK YOU"; T; "TRIES"
240 FOR I = 1 TO 500
250 NEXT T
260 CALL CLEAR
270 GOTO 100
280 PRINT "NOPE, YOU GUESSED WRONG."
290 INPUT "TRY AGAIN"; Y
300 GOTO 160
```

This game is simple, but it has all the components of a playable guessing game. The computer picks a random color in line 120. Line 140 will send the TI-99/4A back for another number if it picks 0. (Zero is not a color code so the computer must get another number.) Lines 130 and 170 keep track of the number of tries that it takes you to guess the color. Line 150 allows you to input a color code and make a guess. Line 190 checks your guess with the color that the TI computer picked. If your guess (Y) equals the computer's selected color (X), lines 200 through 260 will be executed. If Y doesn't equal X, then those lines are skipped and lines 280 through 300 are executed.

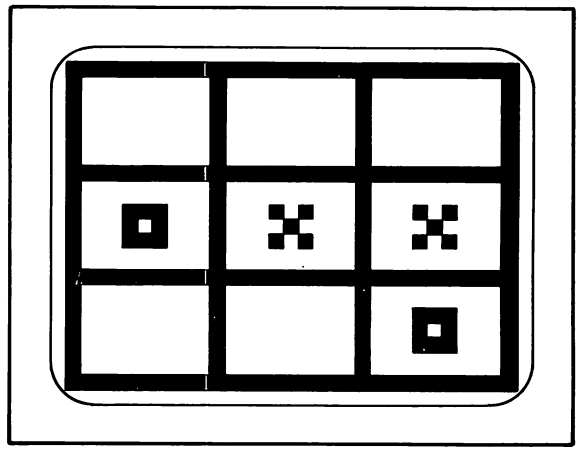
You can change the game to use sounds or numbers. Line 120 will need to be changed. For example, if you want to guess a number from 1 to 100, which the TI computer has selected, change line 120 to read X = INT(100*RND). You will also want to add two lines to tell you if your guess is greater than or less than the computer's number. Those lines would look like this:

```
IF Y < X THEN
IF Y > X THEN
```

The line numbers that follow THEN should send the program to lines that tell you how your guess compares with the computer's number. Try a number guessing game on your own.

TIC-TAC-TOE

Let's try another old favorite, TIC-TAC-TOE. Instead of paper and pencil, you can play on the display screen. Enter the program and, then, we will show you how to play. Here we go.



```
100 CALL CLEAR
110 PRINT "LET'S PLAY TIC-
    TAC-TOE"
120 PRINT
130 PRINT "PRESS THE KEY
    SHOWN IN"
140 PRINT
150 PRINT "EACH BOX"
155 PRINT
156 PRINT "PRESS P FOR A
    NEW GAME"
160 FOR I = 1 TO 1000
170 NEXT I
180 CALL CLEAR
190 GOSUB 1230

200 REM INPUTS ZEROS
210 CALL KEY(0,A,STATUS)
220 IF A<>ASC("Q")
    THEN 270
230 CALL HCHAR(3,5,33,3)
240 CALL VCHAR(4,5,33,2)
250 CALL HCHAR(6,5,33,3)
260 CALL VCHAR(4,7,33,2)
270 IF A<>ASC("W")
    THEN 320
280 CALL HCHAR(3,16,33,3)
```

-Clears the screen.

-Lines 110 to 156 display beginning screen prompts.

-Lines 160 and 170 hold the screen prompts on the screen.

-Branches to 1230 which displays the board.

-Lines 200 to 1200 display the X's and O's depending on the key that is pressed. If key pressed = Q then lines 230 to 260 will plot an O in block 1. If Q is not pressed, the computer goes to 270. This is the playing loop. The computer cycles through until a key is pressed and, then, displays either an X or an O in the correct block.

```
290 CALL VCHAR(4,16,33,2)
300 CALL HCHAR(6,16,33,3)
310 CALL VCHAR(4,18,33,2)
320 IF A<>ASC("E") THEN
    370
330 CALL HCHAR(3,26,33,3)
340 CALL VCHAR(4,26,33,2)
350 CALL HCHAR(6,26,33,3)
360 CALL VCHAR(4,28,33,2)
370 IF A<>ASC("A")
    THEN 420
380 CALL HCHAR(10,5,33,3)
390 CALL VCHAR(11,5,33,2)
400 CALL  HCHAR(13,5,33,3)
410 CALL VCHAR(11,7,33,2)
420 IF A<>ASC("S") THEN
    470
430 CALL HCHAR(10,16,33,2)
440 CALL VCHAR(11,16,33,2)
450 CALL HCHAR(13,16,33,3)
460 CALL VCHAR(11,18,33,2)
470 IF A<>ASC("D")
    THEN 520
480 CALL HCHAR(10,26,33,3)
490 CALL VCHAR(11,26,33,2)
500 CALL HCHAR(13,26,33,3)
510 CALL VCHAR(11,28,33,2)
520 IF A<>ASC("Z") THEN
    570
530 CALL HCHAR(18,5,33,3)
540 CALL VCHAR(19,5,33,2)
550 CALL HCHAR(21,5,33,3)
560 CALL VCHAR(19,7,33,2)
570 IF A<>ASC("X") THEN
    620
580 CALL HCHAR(18,16,33,3)
590 CALL VCHAR(19,16,33,2)
600 CALL HCHAR(21,16,33,3)
610 CALL VCHAR(19,18,33,2)
620 IF A<>ASC("C")
    THEN 670
```

```
630 CALL HCHAR(18,26,33,3)
640 CALL VCHAR(19,26,33,2)
650 CALL HCHAR(21,26,33,
    3)
660 CALL VCHAR(19,28,33,
    2)
670 IF A<>ASC("R")          -Begins X's.
    THEN 730
680 CALL HCHAR(3,5,33)
690 CALL HCHAR(5,5,33)
700 CALL HCHAR(3,7,33)
710 CALL HCHAR(5,7,33)
720 CALL HCHAR(4,6,33)
730 IF A<>ASC("T")
    THEN 790
740 CALL HCHAR(3,16,33)
750 CALL HCHAR(5,16,33)
760 CALL HCHAR(3,18,33)
770 CALL HCHAR(5,18,33)
780 CALL HCHAR(4,17,33)
790 IF A<>ASC("Y")
    THEN 850
800 CALL HCHAR(3,26,33)
810 CALL HCHAR(5,26,33)
820 CALL HCHAR(3,28,33)
830 CALL HCHAR(5,28,33)
840 CALL HCHAR(4,27,33)
850 IF A<>ASC("F")
    THEN 910
860 CALL HCHAR(11,5,33)
870 CALL HCHAR(13,5,33)
880 CALL HCHAR(11,7,33)
890 CALL HCHAR(13,7,33)
900 CALL HCHAR(12,6,33)
910 IF A<>ASC("G")
    THEN 970
920 CALL HCHAR(11,16,33)
930 CALL HCHAR(13,16,33)
940 CALL HCHAR(11,18,33)
950 CALL HCHAR(13,18,33)
```

```
960  CALL HCHAR(12,17,33)
970  IF A<>ASC("H")
     THEN 1030
980  CALL HCHAR(11,26,33)
990  CALL HCHAR(13,26,33)
1000 CALL HCHAR(11,28,33)
1010 CALL HCHAR(13,28,33)
1020 CALL HCHAR(12,27,33)
1030 IF A<>ASC("V")
     THEN 1090
1040 CALL HCHAR(19,5,33)
1050 CALL HCHAR(21,5,33)
1060 CALL HCHAR(19,7,33)
1070 CALL HCHAR(21,7,33)
1080 CALL HCHAR(20,6,33)
1090 IF A<>ASC("B")
     THEN 1150
1100 CALL HCHAR(19,16,33)
1110 CALL HCHAR(21,16,33)
1120 CALL HCHAR(19,18,33)
1130 CALL HCHAR(20,17,33)
1140 CALL HCHAR(21,18,33)
1150 IF A<>ASC("N") TH
     EN 1210
1160 CALL HCHAR(19,26,33)
1170 CALL HCHAR(21,26,33)
1180 CALL HCHAR(19,28,33)
1190 CALL HCHAR(21,28,33)
1200 CALL HCHAR(20,27,33)
1210 IF A<>ASC("P") TH      -Clears the board for a new game
     EN 210                  if P is pressed.
1220 GOTO 180
1230 CALL CHAR(33,"FFFFFF   -Defines character 33 as a solid
     FFFFFFFFFF")            block.
1240 REM PLOTS THE          -Lines 1240 to 1320 plot the play-
     BOARD                   ing board.
1250 CALL VCHAR(1,2,33,24)
1260 CALL HCHAR(1,2,33,31)
1270 CALL VCHAR(1,12,33,
     24)
```

```
1280 CALL VCHAR(1,22,33,
     24)
1290 CALL VCHAR(1,32,33,
     23)
1300 CALL HCHAR(24,1,33,
     31)
1310 CALL HCHAR(8,2,33,31)
1320 CALL HCHAR(16,2,33,
     31)
1330 REM PLOTS X PROMPTS
1340 CALL HCHAR(7,3,88)
1350 CALL HCHAR(7,4,61)
1360 CALL HCHAR(7,5,82)
1370 CALL HCHAR(7,13,88)
1380 CALL HCHAR(7,14,61)
1390 CALL HCHAR(7,15,84)
1400 CALL HCHAR(7,23,88)
1410 CALL HCHAR(7,24,61)
1420 CALL HCHAR(7,25,89)
1430 CALL HCHAR(15,3,88)
1440 CALL HCHAR(15,4,61)
1450 CALL HCHAR(15,5,70)
1460 CALL HCHAR(15,13,88)
1470 CALL HCHAR(15,14,61)
1480 CALL HCHAR(15,15,71)
1490 CALL HCHAR(15,23,88)
1500 CALL HCHAR(15,24,61)
1510 CALL HCHAR(15,25,72)
1520 CALL HCHAR(23,3,88)
1530 CALL HCHAR(23,4,61)
1540 CALL HCHAR(23,5,86)
1550 CALL HCHAR(23,13,88)
1560 CALL HCHAR(23,14,61)
1570 CALL HCHAR(23,15,66)
1580 CALL HCHAR(23,23,88)
1590 CALL HCHAR(23,24,61)
1600 CALL HCHAR(23,25,78)
1610 REM PLOTS THE O PRO
     MPTS
1620 CALL HCHAR(7,9,79)
1630 CALL HCHAR(7,10,61)
```

-Lines 1340 to 1600 display the (X=__) prompts on the playing board.

-Lines 1610 to 1880 display the (O=__) prompts on the playing board.

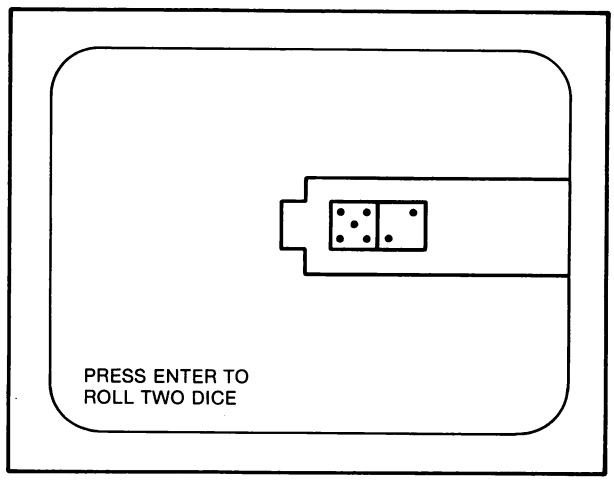
```
1640 CALL HCHAR(7,11,81)
1650 CALL HCHAR(7,19,79)
1660 CALL HCHAR(7,20,61)
1670 CALL HCHAR(7,21,87)
1680 CALL HCHAR(7,28,79)
1690 CALL HCHAR(7,29,61)
1700 CALL HCHAR(7,30,69)
1710 CALL HCHAR(15,9,79)
1720 CALL HCHAR(15,10,61)
1730 CALL HCHAR(15,11,65)
1740 CALL HCHAR(15,19,79)
1750 CALL HCHAR(15,20,61)
1760 CALL HCHAR(15,21,83)
1770 CALL HCHAR(15,28,79)
1780 CALL HCHAR(15,29,61)
1790 CALL HCHAR(15,30,68)
1800 CALL HCHAR(23,9,79)
1810 CALL HCHAR(23,10,61)
1820 CALL HCHAR(23,11,90)
1830 CALL HCHAR(23,19,79)
1840 CALL HCHAR(23,20,61)
1850 CALL HCHAR(23,21,88)
1860 CALL HCHAR(23,28,79)
1870 CALL HCHAR(23,29,61)
1880 CALL HCHAR(23,30,67)
1890 RETURN
```

–Returns to line 200 and play can begin.

The blocks on the playing field are numbered from one to nine beginning with the top left block and moving to the right. There are prompts in each block to tell you which keys to press. When someone wins, press the letter P and you can play again. Our kids think this version is more fun than the "old way." It also helps the little ones recognize the letters of the alphabet.

TED AND DAVE'S CASINO

How about a little gambling for you older folks. We call this next game "TED AND DAVE'S CASINO." At your command, the TI computer rolls the dice. You begin with $10.00. If you roll a seven or eleven, you



win $2.00. If you roll a twelve or less than four, you lose $2.00. If any other number comes up, the house takes $1.00 for the roll. You can raise the stakes, if you like, by changing lines 1290, 1400, and 1470. You can also change which numbers win and lose by changing the sums of DL and DR in lines 1010 to 1030. Enter the program, read our line descriptions, and then experiment.

```
100 CALL CLEAR
110 M = 10
120 GOSUB 1100
130 RANDOMIZE
140 CALL CLEAR
150 PRINT "(12 spaces) WEL
    COME"
160 PRINT
170 PRINT "(14 spaces) TO"
180 PRINT
190 PRINT "(8 spaces) TED
    AND DAVE'S"
200 PRINT
210 PRINT "(12 spaces) CAS
    INO"
220 FOR I = 1 TO 1000
230 NEXT I
```

-Clears the screen.
-Gives you $10.00 to begin with.
-Branches to line 1100.
-TI computer randomizes for dice rolls.
-Lines 150 to 500 display the beginning screen prompts.

```
240 CALL CLEAR
250 PRINT "THIS IS A DICE
    GAME"
260 PRINT
270 PRINT "PRESS ENTER TO
    ROLL THE DICE"
280 PRINT
290 PRINT "YOU BEGIN WI
    TH $10.00"
300 PRINT
310 PRINT "IF YOU ROLL A 7
    OR 11"
320 PRINT
330 PRINT "YOU WIN $2"
340 PRINT
350 PRINT "IF YOU ROLL LE
    SS THAN 4"
360 PRINT
370 PRINT "OR A 12 YOU
    LOSE $2"
380 PRINT
390 PRINT "IF YOU ROLL
    ANY OTHER NUMBER"
400 PRINT
410 PRINT "THE HOUSE TA
    KES $1"
420 PRINT
430 PRINT "FOR THE ROLL"
440 FOR I = 1 TO 2500
450 NEXT I
460 CALL CLEAR
470 PRINT "(5 spaces) TRY
    YOUR LUCK"
480 FOR Q = 1 TO 500
490 NEXT Q
500 CALL CLEAR
510 READ F
520 IF F = 999 THEN 590
530 CALL SOUND(350,F,0)
540 GOTO 510
```

-Lines 510 to 570 play the beginning song.

```
550 DATA 262,330,330,392,
    392
560 DATA 523,523,659,659,
    523,523,392,392,330
570 DATA 330,999
580 CALL CLEAR
590 PRINT "TO ROLL TWO
    DICE"
600 INPUT "PRESS ENTER":
    KY$
610 CALL HCHAR(7,12,111,
    21)
620 CALL HCHAR(13,12,111,
    21)
630 CALL VCHAR(9,10,111,3)
640 CALL VCHAR(8,11,111)
650 CALL VCHAR(12,11,111)
660 CALL VCHAR(9,20,110)
670 CALL VCHAR(11,20,109)
680 CALL VCHAR(10,19,108)
690 CALL VCHAR(10,21,107)
700 CALL VCHAR(9,16,110)
710 CALL VCHAR(11,16,109)
720 CALL VCHAR(10,15,108)
730 CALL VCHAR(10,17,107)
740 GOSUB 1050
750 IF DR<>1 THEN 770
760 CALL VCHAR(10,20,101)
770 IF DR<>2 THEN 790
780 CALL VCHAR(10,20,102)
790 IF DR<>3 THEN 810
800 CALL VCHAR(10,20,103)
810 IF DR<>4 THEN 830
820 CALL VCHAR(10,20,104)
830 IF DR<>5 THEN 850
840 CALL VCHAR(10,20,105)
850 IF DR<>6 THEN 870
860 CALL VCHAR(10,20,106)
870 IF DL<>1 THEN 890
880 CALL VCHAR(10,16,101)
890 IF DL<>2 THEN 910
```

-Lines 590 and 600 give prompt to roll dice and allow input from enter key.

-Lines 610 to 650 display the dice table.

-Lines 660 to 690 display lines that frame the die dots on the right die.

-Lines 700 to 730 display lines that frame the die dots on the left die.

-Branches to 1050.

-Lines 750 to 860 condition and display the numbers on the right die.

-Lines 870 to 980 condition and display the numbers on the left die.

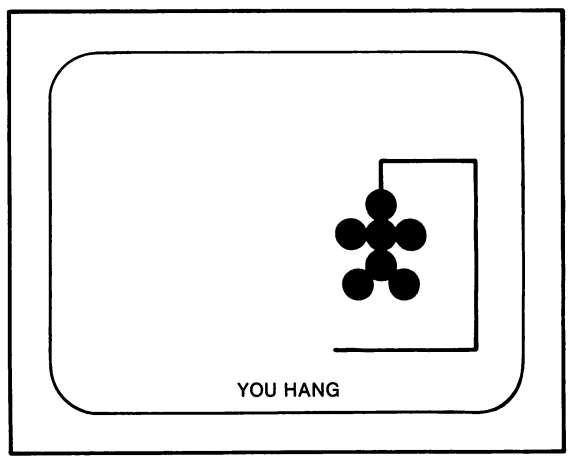
```
 900 CALL VCHAR(10,16,102)
 910 IF DL <> 3 THEN 930
 920 CALL VCHAR(10,16,103)
 930 IF DL <> 4 THEN 950
 940 CALL VCHAR(10,16,104)
 950 IF DL <> 5 THEN 970
 960 CALL VCHAR(10,16,105)
 970 IF DL <> 6 THEN 990
 980 CALL VCHAR(10,16,106)
 990 FOR HOLD = 1 TO 10
     00
1000 NEXT HOLD
1010 IF (DL+DR=7) + (DL+     -Lines 1010 to 1040 are the scor-
     DR=11) THEN 1340          ing conditions.
1020 IF (DL+DR < 4) + (DL+
     DR=12) THEN 1260
1030 IF (DL+DR=4) + (DL+
     DR=5) + (DL+DR=6) +
     (DL+DR=8) + (DL+DR=
     9) + (DL+DR=10) THEN
     1450
1040 GOTO 140
1050 DL = INT(7*RND)          -Computes the value of the left
1060 IF DL< 1 THEN 1050        die.
1070 DR= INT(7*RND)           -Computes the value for the right
1080 IF DR< 1 THEN 1070        die.
1090 RETURN
1100 CALL SCREEN(7)           -Colors the screen red.
1110 FOR SET = 9 TO 10        -Lines 1110 to 1130 color the
1120 CALL COLOR(SET,2,7)       characters black on red.
1130 NEXT SET
1140 CALL CHAR(101,"00000     -Displays dice display #1.
     01818000000")
1150 CALL CHAR(102,"00060     -Displays dice display #2.
     60000606000")
1160 CALL CHAR(103,"00060     -Displays dice display #3.
     61818606000")
1170 CALL CHAR(104,"00666     -Displays dice display #4.
     60000666600")
1180 CALL CHAR(105,"00666     -Displays dice display #5.
     61818666600")
```

```
1190 CALL CHAR(106,"66660
     06666006666")          -Displays dice display #6.
1200 CALL CHAR(107,"80808   -Lines 1200 to 1230 define the
     08080808080")            thin line that frames the dice.
1210 CALL CHAR(108,"01010
     10101010101")
1220 CALL CHAR(109,"FF000
     00000000000")
1230 CALL CHAR(110,"00000
     000000000FF")
1240 CALL CHAR(111,"FFFFF   -Character that makes the dice
     FFFFFFFFFF")             table.
1250 RETURN
1260 PRINT "YOU LOSE"       -Lose condition.
1270 CALL SOUND(200,392,    -Lines 1270 and 1280 are the lose
     0)                       sound.
1280 CALL SOUND(200,262,
     1)
1290 M = M - 2
1300 PRINT "YOU HAVE $ ;    -Lose score computation.
     M
1310 FOR I = 1 TO 800       -Lines 1310 and 1320 hold the
1320 NEXT I                   prompt on the screen.
1330 GOTO 580               -Loops back to line 580.
1340 PRINT "YOU WIN"        -Win prompt.
1350 CALL SOUND(300, 262,   -Lines 1350 to 1400 call the "Win"
     0)                       song.
1360 CALL SOUND(300,294,
     0)
1370 CALL SOUND(300,330,
     0)
1380 CALL SOUND(300,294,
     0)
1390 CALL SOUND(300,330,
     0)
1400 CALL SOUND(300,262,
     0)
1410 M = M + 2              -Computer win score.
1420 PRINT "YOU HAVE $ ;    -Prints score.
     M
```

```
1430 FOR I = 1 TO 800          -Lines 1430 and 1440 hold the
1440 NEXT I                      prompt on the screen.
1450 GOTO 580                  -Loops back to 580.
1460 PRINT "NOT A WIN          -Neutral sound.
     NER, HOUSE COLLE
     CTS ONE DOLLAR"
1470 CALL SOUND(200, -6,       -Neutral sound.
     1)
1480 M = M - 1                 -Computes neutral sound.
1490 PRINT "YOU HAVE $ ;       -Prints score.
     M
1500 FOR I = 1 TO 800          -Lines 1500 and 1510 hold prompt
1510 NEXT I                      on screen.
1520 GOTO 580                  -Loops back to line 580.
```

When you are tired of losing your shirt rolling dice, try a nice quiet game of Hang Man.

HANG MAN



Player Number 1 tells the TI computer how many letters are in the secret word. Then, one at a time, type in the letters of the word by typing a letter and, then, pressing ENTER. You have ten chances to guess the word your opponent entered. If you don't guess correctly, You Hang. Enter the following program.

```
100 CALL CLEAR
110 GOSUB 680
120 CALL CLEAR
130 INPUT "HOW MANY LE
    TTERS ARE IN YOUR SEC
    RET WORD": L
140 FOR X = 1 TO L
150 INPUT C$

160 Y(X) = ASC(C$)
170 Z(X) = Y(X)
180 NEXT X
190 FOR J = 1 TO 10
200 CALL CLEAR
210 PRINT
220 PRINT
230 FOR A = 1 TO L
240 PRINT Y$
250 IF Y(A) = Z(A) THEN 640
    ELSE 260
260 IF Y(A)<>Z(A) THEN
    660 ELSE 270
270 NEXT A
280 PRINT
290 PRINT
300 H = 0
310 PRINT Y$; "ENTER GUE
    SS NO."; J
320 PRINT
330 INPUT C$
340 F = ASC(C$)
350 FOR G = 1 TO L
360 IF Z(G)<>F THEN 380
370 Z(G) = 0
380 IF Z(G)<>0 THEN 400
390 H = H + 1
400 NEXT G
410 IF H = L THEN 480
420 NEXT J
430 PRINT
```

-Clears the screen.
-Branches to line 680 to plot the beginning prompts.
-Allows Player 1 to tell the computer the number of letters in the word.
-Number of letters in the word.
-Allows Player 1 to type in the letters.
-Defines the value of B(A) as the inputted character.
-Guess = 1 to 10.
-Number of guessed letters.
-This will plot a space for each letter.
-Plots the letter if guessed correctly.
-Screen prompt for number of guesses.
-Player 2 inputs guessed letter.
-F = guessed letter.
-Lines 360 to 380 test and compute the number of guesses.
-Score device to compute the number of guesses.
-Correct word guess condition.

```
440 GOSUB 1500
450 PRINT
460 PRINT Y$; "THE WORD
    WAS"
470 GOTO 520
480 REM CORRECT WORD
    GUESS
490 PRINT
500 PRINT Y$; "YOU GOT IT
    IN"; J
510 PRINT
520 FOR Q = 1 TO L
530 PRINT CHR$(Y(Q));
540 NEXT Q
550 PRINT
560 PRINT Y$; "CARE TO
    TRY AGAIN"
570 INPUT C$
580 IF ASC(C$) = ASC("Y")
    THEN 120
590 PRINT Y$ "SEE YOU LA
    TER";
600 FOR I = 1 TO 100
610 NEXT I
620 PRINT
630 GOTO 590
640 PRINT "_____";
650 GOTO 260
660 PRINT CHR$(Y(A));
670 GOTO 270
680 CALL CHAR(96,"FFFFFFF
    FFFFFFFFF")
690 CALL CHAR(104,"0F0F0
    F0F0F0F0F0F")
700 CALL CHAR(112,"FFFFF
    FFF00000000")
710 CALL CHAR(120,"F0F0F
    0F0F0F0F0F0")
720 CALL CHAR(128,"3C7E
    FFFFFFFF7E3C")
```

-Branches to Hanged Man subroutine.

-Terminal guess prompt for wrong word.

-Guess prompts for correct word.

-Prints the word and the number of guesses it took you to guess it.

-Allows you to try again.

-Lines 570 and 580 loop the program back to play again if you type in Y.

-If Y isn't pressed, the game ends.

-Holds the prompt on the screen.

-Lines 640 and 650 plot the _____ and returns.

-Plots the character at the beginning.

-Lines 680 to 730 define characters for the Hang Man block letters and gallows.

```
730  CALL CHAR(113,"00000
     000FFFFFFFF")
740  REM BLOCK LETTERS
750  CALL VCHAR(3,3,96,4)      -Lines 750 to 780 are the H.
760  CALL VCHAR(3,6,96,4)
770  CALL HCHAR(4,4,113,2)
780  CALL HCHAR(5,4,112,2)
790  CALL VCHAR(3,9,96,4)      -Lines 790 to 820 are the A.
800  CALL VCHAR(3,11,96,4)
810  CALL VCHAR(3,10,96)
820  CALL VCHAR(5,10,96)
830  CALL VCHAR(3,14,96,4)     -Lines 830 to 870 are the N.
840  CALL VCHAR(3,18,96,4)
850  CALL VCHAR(3,15,96)
860  CALL VCHAR(4,16,96)
870  CALL VCHAR(5,17,96)
880  CALL VCHAR(3,21,96,4)     -Lines 880 to 910 are the G.
890  CALL HCHAR(3,22,96,2)
900  CALL HCHAR(6,22,96,2)
910  CALL HCHAR(5,23,96)
920  CALL VCHAR(9,14,96,4)     -Lines 920 to 960 are the M.
930  CALL VCHAR(9,18,96,4)
940  CALL VCHAR(9,15,96)
950  CALL VCHAR(9,17,96)
960  CALL VCHAR(10,16,96)
970  CALL VCHAR(9,21,96,4)     -Lines 970 to 1000 are the A.
980  CALL VCHAR(9,23,96,4)
990  CALL VCHAR(9,22,96)
1000 CALL VCHAR(11,22,96)
1010 CALL VCHAR(9,26,96,4)     -Lines 1010 to 1050 are the N.
1020 CALL VCHAR(9,30,96,4)
1030 CALL VCHAR(9,27,96)
1040 CALL VCHAR(10,28,96)
1050 CALL VCHAR(11,29,96)
1060 FOR I = 1 TO 400
1070 NEXT I
1080 TONE = 226                -Lines 1080 to 1120 are the Musi-
1090 FOR SCALE = 1 TO 8         cal Scale Approximation subrou-
1100 CALL SOUND(500,TO          tine from Chapter 4.
     NE,1)
```

```
1110 TONE = TONE + 18
1120 NEXT SCALE
1130 PRINT "HERE IS HOW
     YOU PLAY"
1140 PRINT
1150 PRINT "PLAYER 1 ENT
     ERS THE NUMBER OF"
1160 PRINT
1170 PRINT "LETTERS IN THE
     SECRET WORD,"
1180 FOR I = 1 TO 1000
1190 NEXT I
1200 PRINT
1210 PRINT "THEN PLAYER 1
     TYPES IN EACH"
1220 PRINT
1230 PRINT "LETTER ONE AT
     A TIME."
1240 PRINT
1250 PRINT "BE SURE THAT
     PLAYER 2"
1260 FOR I = 1 TO 1000
1270 NEXT I
1280 PRINT
1290 PRINT "DOESN'T PEEK,
     AND THAT"
1300 PRINT
1310 PRINT "ENTER IS PRESS
     ED AFTER EACH LET
     TER"
1320 PRINT
1330 PRINT
1340 PRINT "PLAYER 2 GUES
     SES LETTERS BY"
1350 PRINT
1360 PRINT "TYPING THE LE
     TTER AND ENTER"
1370 PRINT
1380 FOR I = 1 TO 1000
1390 NEXT I
```

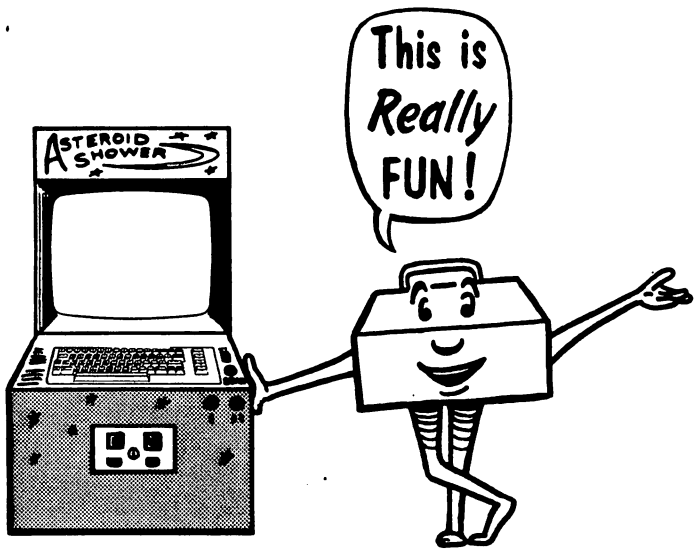
-Lines 1130 to 1490 display the beginning screen prompts.

```
1400 PRINT "PLAYER 2 HAS
     TEN TRYS TO GUESS
     THE WORD"
1410 PRINT
1420 PRINT "IF YOU DON'T
     GET THE WORD IN TE
     N TRYS"
1430 PRINT
1440 PRINT "YOU HANG"
1450 PRINT
1460 PRINT "LET'S PLAY"
1470 FOR I = 1 TO 1000
1480 NEXT
1490 RETURN
1500 CALL CLEAR                    -Lose condition.
1510 CALL SOUND(500,262,           -Lines 1510 to 1610 play the song
     1)                             that you hear when you lose.
1520 CALL SOUND(500,262,
     1)
1530 CALL SOUND(500,262,
     1)
1540 CALL SOUND(500,262,
     1)
1550 CALL SOUND(500,311,
     1)
1560 CALL SOUND(500,294,
     1)
1570 CALL SOUND(500,294,
     1)
1580 CALL SOUND(500,262,
     1)
1590 CALL SOUND(500,262,
     1)
1600 CALL SOUND(500,247,
     1)
1610 CALL SOUND(500,262,
     1)
1620 CALL VCHAR(9,25,120,          -Lines 1620 to 1720 plot the Hang
     10)                            Man.
1630 CALL HCHAR(9,20,112,
     5)
```

```
1640 CALL VCHAR(9,19,104,
     3)
1650 CALL VCHAR(12,19,128,
     3)
1660 CALL VCHAR(13,18,128)
1670 CALL VCHAR(13,20,128)
1680 CALL VCHAR(15,18,128)
1690 CALL VCHAR(15,20,128)
1700 PRINT "YOU HANG"
1710 FOR I = 1 TO 800
1720 NEXT I
1730 RETURN
```

Player 1 must select a word with no more than ten letters, otherwise, any letters in any combinations can be used. Remember that the TI computer doesn't know how to spell. If a word is misspelled when it is entered, the same misspelling must occur to score a correct guess. if you like, you can enter the word backwards and play backwards Hang Man.

We hope you have enjoyed the games in this chapter. Study how we have done things, like scoring and setting the right and wrong answer conditions. These concepts will be used again in the next chapter. Use our games as models and try some games of your own. Try Blackjack, or maybe another type of matching or guessing game. Begin with a simple idea and then dress it up some. When you are ready, we will move into the exciting world of *Arcade Games.*



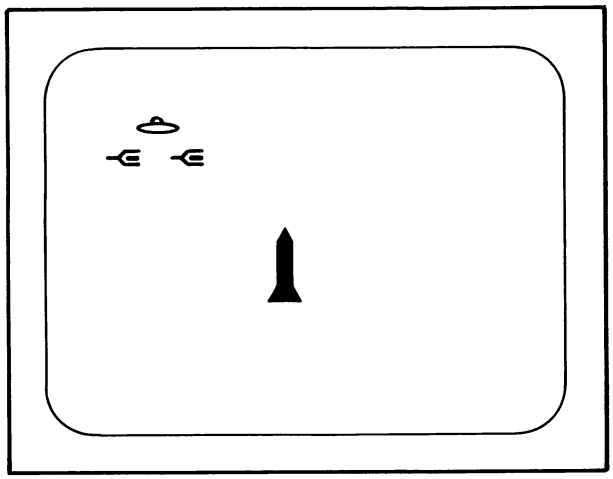
Arcade-Type Game Programs 10

Creating arcade games can be a lot of fun, and can be extremely frustrating if you are not careful. The best advice we can offer is to begin with a very simple game. Then, take the game apart one piece at a time and build your game step-by-step. That is exactly the way we did the games in this chapter. Don't get too fancy at first. If you do, you will get extremely frustrated.

Now for one other piece of advice; if you intend to do a lot of work with arcade games, you should buy the TI Extended BASIC module. All of the games in this chapter could be a lot fancier if we had used Extended BASIC.

Let's try a very simple arcade game first.

SAUCER BLASTER



```
100 CALL CLEAR
110 CALL SCREEN(5)
120 PRINT "(6 spaces) SAU
    CER BLASTER"
130 PRINT
140 PRINT
150 PRINT "PRESS ANY KEY
    TO FIRE"
160 PRINT
170 PRINT "AIM AT THE SAU
    CER"
```

-Clears the screen.
-Calls the screen to dark blue.
-Hit SPACEBAR 6 times. Do not type in the words (6 spaces).

-Lines 120 to 180 display the title and instructions.

```
180 PRINT "DON'T HIT THE
    ESCORT SHIPS"
190 FOR I = 1 TO 1000          -Lines 190 and 200 hold the in-
200 NEXT I                      structions on the screen.
210 CALL CLEAR                 -Clears the instructions.
220 CALL SCREEN(16)            -Calls the screen white.
230 FOR I = 1 TO 400           -Lines 230 and 240 hold the white
240 NEXT I                      screen.
250 SCORE = 0                  -Defines the score variable.
260 BAD = 0                    -Defines the score variable if you
                                hit the escort ships.
270 CALL SCREEN(5)             -Calls the screen to dark blue.
280 CALL COLOR(10,7,5)         -Colors the missile.
290 CALL COLOR(9,2,5)          -Colors escort ships.
300 CALL COLOR(11,2,5)         -Colors the saucer.
310 FOR SET = 3 TO 8
320 CALL COLOR(SET,2,16)
330 NEXT SET                   -Colors all character sets from 3
340 CALL CLEAR                  to 8 (all the letters and numbers).
350 CALL CHAR(104,"00183       -Defines the missile graphic
    CFFFF180000")               character.
360 CALL CHAR(112,"3C3C3       -Defines the saucer graphic char-
    C3C7E18183C")               acter.
370 CALL CHAR(96,"0F1020       -Defines the escort ship graphic
    CFCF20100F")                character.
380 CALL SOUND(4250,-5,0)      -Calls saucer sound.
390 FOR X = 3 TO 30            -Variable for displaying escort
400 CALL CLEAR                  ship and saucer.
410 CALL HCHAR(2,X,104)        -Displays saucer.
420 CALL HCHAR(4,X+2, 96)      -Lines 420 and 430 display the
430 CALL HCHAR(4,X-2, 96)       escort ships.
440 CALL KEY(0,KEY,R)          -Lines 440 to 460 call key condi-
450 IF R = 0 THEN 470           tion 1 = Fire.
460 IF R = 1 THEN 600
470 NEXT X
480 CALL SOUND(4250,-5,0)      -Saucer sound coming back a-
                                cross the screen.
490 FOR X = 30 TO 3 STEP -1    -Condition for the saucer to
                                come back across the screen.
500 CALL KEY(0,KEY,R)          -Lines 500 to 520 call key condi-
510 IF R = 0 THEN 530           tion 1 = Fire.
```

```
520 IF R = 1 THEN 600
530 CALL CLEAR
540 CALL HCHAR(2,X,104)          -Lines 540 to 560 display saucer
550 CALL HCHAR(4,X+2,96)           and escorts coming back across
560 CALL HCHAR(4,X-2,96)           the screen.
570 NEXT X
580 CALL SOUND(4250,-5,0)        -More saucer sounds.
590 GOTO 390                     -Loop to line 390 to redisplay
                                   saucer.
600 FOR Y = 22 TO 1 STEP         -Variable for missile trajectory.
    -5
610 CALL CLEAR
620 CALL VCHAR(Y,15,112)         -Lines 620 and 630 plot the
630 NEXT Y                         missile.
640 IF X = 15 THEN 670           -Hit condition sends TI computer
                                   to line 670.
650 IF(X + 2 = 15) + (X - 2 =    -Hit escort ships condition.
    15) THEN 750
660 GOTO 470
670 CALL CLEAR
680 CALL SOUND(4000,1000,        -Hit sound.
    0)
690 PRINT "DIRECT HIT"           -Hit prompt.
700 SCORE = SCORE +10            -Lines 700 and 710 are the calcu-
710 PRINT SCORE; "SCORE"           lated positive scoring values.
720 FOR Q = 1 TO 800             -Holds score and direct hit on the
730 NEXT Q                         screen.
740 GOTO 340
750 CALL CLEAR
760 CALL SOUND(1000,500,         -Lines 760 and 770 are the hit
    1)                             sound and prompt for the escort
770 PRINT "YOU HIT THE             ships.
    WRONG SHIP"
780 SCORE = 0                    -Negative score calculation.
790 PRINT BAD; "SCORE"
800 FOR Q = 1 TO 800             -Holds negative score and prompt
810 NEXT Q                         on the screen.
820 GOTO 340
```

The object of Saucer Blaster is to shoot the saucer without hitting the escort ships. You get 10 points every time you hit the saucer. But you

lose all of your points if you hit either of the escort ships. You can only shoot from one position.

This program has all of the elements of an arcade game. It has graphics, animation, color, sound, and a scoring mechanism. If you study Saucer Blaster, you will see ways to change some of the lines and thus create an entirely different game. After you study each line, try these changes. These changes will allow you to play a game that fires horizontally instead of vertically. We will give you the new variables plus some advice and you can create your own game.

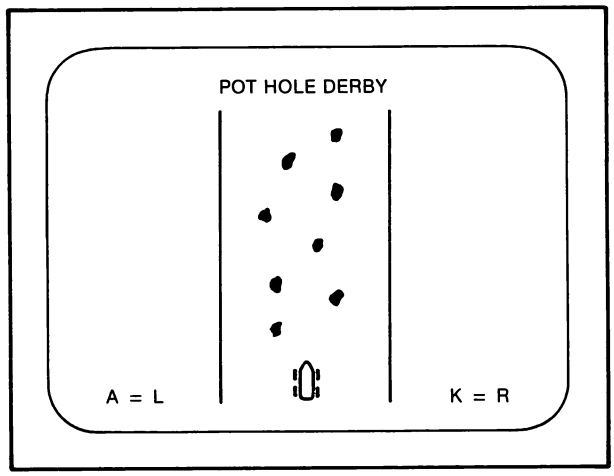
Lines 120—200. Change the prompts to describe or title your game.
Line 220. Change color as desired.
Lines 270—330. Change colors if you like.
Lines 350—370. Define your own characters.
Line 390. FOR X=3 TO 22.
Line 410. CALL HCHAR (X,2,104).
Line 420. CALL HCHAR (X+2,4,96).
Line 430. CALL HCHAR (X−2, 4,96).
Line 490. FOR X=22 TO 3 STEP −1.
Line 540. CALL HCHAR (X,2,104).
Line 550. CALL HCHAR (X+2,4,96).
Line 560. CALL HCHAR (X−2,4,96).
Line 600. FOR Y=32 TO 1 STEP −5.
Line 620. CALL HCHAR (10,Y,112).
Line 640. IF X = 10 THEN 670.
Line 650. IF (X + 2 = 10) + (X − 2 = 10) THEN 760.
Lines 690, 770, and 790. Can be changed to read anyway that you like.

All of the CALL SOUND lines can be changed to fit your game. You can learn a lot by working with this program and designing your own game.

Since this game uses defined characters, you can make some minor changes in the characters and the game will look entirely different. Now let's look at a more complicated arcade game.

POT HOLE DERBY

The purpose of the game is to drive your car to the top of the screen without hitting a pothole or running off the road. The TI computer



controls your forward motion, but you can turn left by pressing the L key and right by pressing the K key. You score 50 points if you can make it to the top of the screen and you get to try again. When you crash, your score goes to ZERO and you must begin again.

The complete program is listed later, but here is how we start. First, we put the car on the screen so that we can control the left and right directions.

```
 50 CALL CLEAR
 90 A = 96                          -Car variable as defined in line
                                     120.
100 Z = 24                          -Starting row of car.
110 COL = 16                        -Starting column of car.
120 CALL CHAR(96,"5A7E5            -Defines car.
    A18185A7E5A")
130 CALL HCHAR(Z,COL,32)           -Plots a space to hide the car's
                                     trail.
140 Z = Z - 1                       -Allows the car to move up.
150 IF Z > 4 THEN 170               -If Z is greater than 4, then plot.
160 IF Z < 6 THEN 100               -If Z is less than 6, the car goes
                                     back to the bottom.
170 CALL KEY(0,L,ST)                -Lines 170 to 190 describe key
180 IF (L <> 79) * (L <> 80)         conditions.
    THEN 230
190 IF L <> 79 THEN 220
200 COL = COL -SGN(COL              -Variable computation for the
    -2)                              column.
```

```
210 GOTO 230
220 COL = COL +SGN(30-      -Variable computation for the
    COL)                     column.
230 CALL GCHAR (Z,COL,L)    -Lines 230 and 240 plot the car.
240 CALL HCHAR (Z,COL,A)
250 FOR I = 1 TO 25
260 NEXT I
270 GOTO 130
```

This can be thought of as a subroutine for displaying and controlling a graphics character. If you change the lines 100 and 140 to 160, the car will move down the screen instead of up. Change the following lines so that they look like the following:

```
100 Z = 1
140 Z = Z + 1
150 IF Z < 2 THEN 170
160 IF Z > 22 THEN 100
```

These changes can give a whole new dimension to the game. You will see this in the Asteroid Shower game that we discuss later in this chapter.

Once you have the car in place, you will need to add the potholes and the road. This we did (see complete program for details) and, then, we added a scoring device. Now enter the whole program and see how it plays.

```
100 CALL CLEAR              -Clears the screen.
110 RANDOMIZE               -Randomizes.
120 S = 0                   -Initial value of score.
130 POT = 97                -Pothole character variable.
140 ROAD = 100              -Road character variable.
150 GOSUB 660               -Branches to line 660.
160 CALL SCREEN(15)         -Colors the screen gray.
170 PRINT "(8 spaces) POT   -Lines 170 to 210 display the
    HOLE DERBY"              screen prompts.
180 FOR I = 1 TO 19
190 PRINT
200 NEXT I
210 PRINT "(4 spaces) A = L
    (17 spaces) K = R"::
220 CALL VCHAR (5,11,ROA    -Lines 220 and 230 display the
    D,19)                    road.
```

```
230 CALL VCHAR (5,24,RO
    AD,19)
240 FOR I = 1 TO 25
250 X = INT(RND*12) + 12
260 Y = INT(RND*16) + 7
270 CALL GCHAR(Y,X,L)
280 IF L = POT THEN 250
290 CALL HCHAR(Y,X,POT)
300 NEXT I
310 A = 96
320 ROW =24
330 COL =16
340 CALL HCHAR(ROW,COL
    ,32)
350 ROW = ROW -1
360 RB = 24
370 LB = 11
380 TB = 5
390 IF ROW > 4 THEN 410
400 IF ROW < 6 THEN 320
410 CALL KEY(O,L,ST)
420 IF (L <> 65) * (L <> 75)
    THEN 470
430 IF L <> 65 THEN 460
440 COL = COL - SGN(CO
    L-2)
450 GOTO 470
460 COL = COL + SGN(30
    -COL)
470 CALL GCHAR(ROW,CO
    L,L)
480 IF COL= LB THEN 600
490 IF COL = TB THEN 560
500 IF ROW = TB THEN 560
510 IF L = POT THEN 600
520 CALL HCHAR(ROW, CO
    L,A)
530 FOR I = 1 TO 25
540 NEXT I
```

-Lines 240 to 300 display the potholes at random.

-Car character variable.

-Lines 320 and 330 are the car's beginning coordinates.

-Plots the space that removes the car's trail.

-Right border.

-Left border.

-Top border.

-If row > 4, the program branches to line 410.

-If row < 6, the program branches to line 320.

-Activates the keyboard.

-Lines 420 and 430 define the key condition.

-Lines 440 to 460 move the car left and right.

-Lines 480 and 490 are the crash conditions.

-Score condition.

-Crash condition.

-Plots the car.

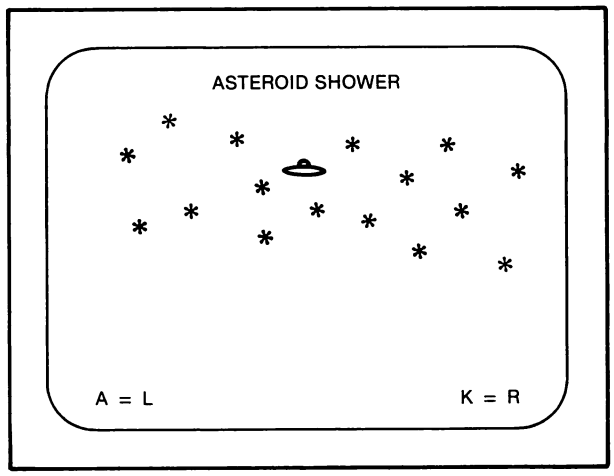
```
550 GOTO 340
560 S = S + 50                    -Computes the score.
570 FOR I = 1 TO 100
580 NEXT I
590 GOTO 520
600 CALL SOUND(500,-7,1)          -Crash sound.
610 FOR I = 4 TO 40 STEP 4
615 NEXT I
620 PRINT "YOU CRASHED"           -Crash prompt.
630 PRINT S; "SCORE"              -Prints the score.
635 FOR A = 1 TO 1000
640 NEXT A
650 GOTO 100
660 CALL CHAR(96,"5A7E5A          -Lines 660 to 680 are the charac-
    18185A7E5A")                   ters defined.
670 CALL CHAR(97,"3C7EFF
    FFFF733C18")
680 CALL CHAR (100,"0F0F0
    F0F0F0F0F")
690 CALL COLOR(10,2,1)
700 FOR SET = 3 TO 8
710 CALL COLOR(SET,15,5)
720 NEXT SET
730 RETURN
```

If you want to change the level of difficulty, increase the value of 25 in line 240 for more potholes and decrease the value for fewer holes. Now let's use the same design to create an entirely different game.

ASTEROID SHOWER

In this game, you are in a flying saucer. The saucer will move from the top of the screen to the bottom by itself, but you control the left and right directional movements with the letter A and K keys. Your mission is to hit as many asteroids as you can, but be careful. If you land to the right of column 16, the blackhole gets you and you lose all your points. Now, let's input the Asteroid Shower.

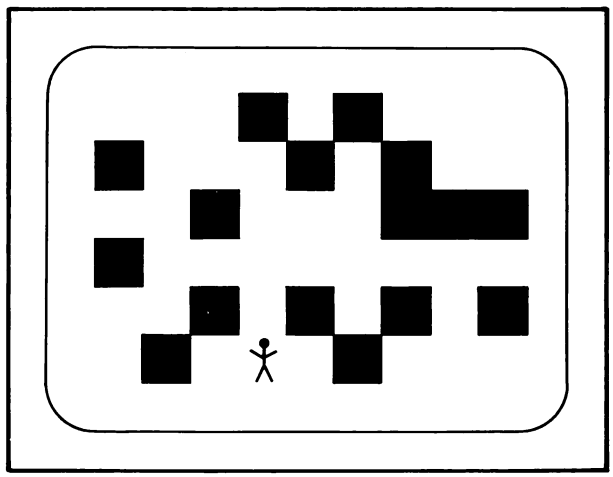
```
100 CALL CLEAR                    -Clears the screen.
110 RANDOMIZE                     -Randomizes.
120 S = 0                         -Initial score.
```




```
130 ASTEROIDS = 104                  -Asteroid character variable.
140 GOSUB 510                        -Branches to line 510.
150 CALL SCREEN(5)                   -Colors the screen dark blue.
160 PRINT "(5 spaces) ASTER          -Lines 160 to 200 display the
    OID SHOWER"                       screen prompts.
170 FOR I = 1 TO 19
180 PRINT
190 NEXT I
200 PRINT "(4 spaces) A = L
    (15 spaces) K = R"
210 FOR I = 1 TO 50                  -Controls the number of aster-
                                      oids.
220 X = INT(RND*30) + 2              -Lines 220 and 230 are the row
                                      and column variables for plot-
230 Y = INT(RND*16) + 6               ting the asteroids.
240 CALL GCHAR(Y,X,L)                -Lines 240 to 270 display the aster-
250 IF L = ASTEROIDS THEN             oids.
    220
260 CALL HCHAR(Y,X,ASTER
    OIDS)
270 NEXT I
280 SAUCER = 96                      -Saucer variable.
290 ROW = 5                          -Initial row value.
300 COL = 18                         -Initial column value.
310 CALL HCHAR(ROW,COL               -Covers the trail of the saucer.
    ,32)
```

```
320 ROW = ROW + 1                    -Keeps the saucer moving down.
330 IF ROW < 2 THEN 350
340 IF ROW > 22 THEN 290             -Takes the saucer back to the top.
350 IF (COL=16)*(ROW=21)             -Blackhole condition.
    THEN 590
360 CALL KEY(0,L,ST)                 -Activates the keyboard.
370 IF (L<>65)*(L<>75)
    THEN 420
380 IF L<>65 THEN 410
390 COL = COL - SGN(C                -Column values. Lines 370 to 390
    OL-2)                             are the key conditions.
400 GOTO 420
410 COL = COL + SGN(30-              -Column values.
    COL)
420 CALL GCHAR(ROW,CO
    L,L)
430 IF L = ASTEROIDS THEN            -Score condition.
    480
440 CALL HCHAR(ROW,COL               -Lines 440 to 460 plot the saucer.
    ,SAUCER)
450 FOR I = 1 TO 25
460 NEXT I
470 GOTO 310
480 CALL SOUND(500,-5,0)             -Calls sound when asteroid is hit.
490 S = S + 10
500 GOTO 440
510 CALL CHAR(96,"00183C             -Defines the saucer character.
    FFFF180000")
520 CALL CHAR (104,"995A3            -Defines asteroid character.
    CFFFF3C5A99")
530 CALL COLOR(10,11,5)              -Colors the saucer black.
540 CALL COLOR(9,2,5)                -Colors the escort ships black.
550 FOR I = 3 TO 8                   -Lines 550 to 570 color all the
560 CALL COLOR(I,16,5)                characters in Sets 3 to 8 white.
570 NEXT I
580 RETURN
590 CALL CLEAR
600 CALL SCREEN(2)                   -Colors the screen black.
610 CALL SOUND(500,-2,0)             -Blackhole sound.
620 PRINT "THE BLACKHO               -Blackhole prompt.
    LE GOT YOU"
```

```
630 PRINT S; "SCORE"          -Prints the score.
640 FOR I = 1 TO 1000
650 NEXT I
660 GOTO 100                  -Returns to allow you to play
                               again.
```



RAT TRAP

The last game in this chapter is called Rat Trap. You are trapped in a maze. You can use the cursor arrows to move up (E), down (X), right (D), and left (S). If you hit a maze wall, the computer will deduct 10 points from your score. Zero is a perfect score.

Here is the program:

```
100 CALL CLEAR                -Clears the screen.
110 M = 97                    -Defines the maze variable.
120 CALL CHAR(97,"FFFFFFF     -Defines the maze character.
    FFFFFFFFF")
130 CALL CHAR(130,"18183      -Defines the man character.
    C5A18242424")
140 CALL SCREEN(16)           -Colors the screen white.
150 FOR SET = 1 TO 8          -Lines 150 to 170 color all the let-
160 CALL COLOR(SET,2,16)       ters and numerals black or white.
170 NEXT SET
180 PRINT "(2 spaces) THIS IS -Lines 180 to 410 print the begin-
    THE RAT TRAP."             ning screen prompts.
```

```
190 PRINT
200 PRINT "YOU MUST GET
    THROUGH THE"
210 PRINT
220 PRINT "TRAP WITHOUT
    HITTING THE"
230 PRINT
240 PRINT "WALLS."
250 PRINT
260 PRINT
270 PRINT "IF YOU HIT A
    WALL, I WILL"
280 PRINT
290 PRINT "DEDUCT 10 POI
    NTS FROM YOUR"
300 PRINT
310 PRINT "SCORE."
320 PRINT
330 PRINT
340 PRINT "USE THE CURS
    OR ARROWS"
350 PRINT
360 PRINT "TO MOVE YOUR
    MAN."
370 PRINT
380 PRINT "(9 spaces) HERE
    WE GO"
390 PRINT
400 PRINT
410 PRINT "(11 spaces) GO
    OD LUCK!"
420 FOR I = 1 TO 2000
430 NEXT I                    -Holds the screen prompts on
                               the screen.
440 SCORE = 0                 -Scoring variable.
450 CALL CLEAR
460 FOR I = 1 TO 200          -Sets the number of maze blocks.
470 RANDOMIZE                 -Computer randomizes the
                               screen location.
480 X = INT(RND*30) +2        -Column variable for maze.
490 Y = INT(RND*20) +2        -Row variable for maze.
500 CALL GCHAR(Y,X,L)         -Lines 500 and 510 plot the maze.
```

```
510 CALL HCHAR(Y,X,M)
520 NEXT I
530 CMAN = 16                    -Initial column value for the man.
540 RMAN = 24                    -Initial row value for the man.
550 OLDCMAN = CMAN               -Lines 550 and 560 state the value
560 OLDRMAN = RMAN                of the next display of the man.
570 CALL KEY(0,A,STATUS)         -Lines 570 to 650 allow us to move
580 IF A<>ASC("E") THEN           the man.
    600
590 RMAN = RMAN-SGN(R
    MAN-1)
600 IF A<>ASC("X") THEN
    620
610 RMAN = RMAN + SGN
    (22-RMAN)
620 IF A<>ASC("D")
    THEN 640
630 CMAN = CMAN + SGN
    (31-CMAN)
640 IF A<>ASC("S") THEN
    660
650 CMAN = CMAN - SGN
    (CMAN-2)
660 IF(RMAN = OLDRMAN)
    * (CMAN = OLDCMAN)
    THEN 710
670 CALL GCHAR(RMAN,CM           -Plots the first man.
    AN,A)
680 IF A = M THEN 730            -Hit condition.
690 IF RMAN = 1 THEN 770         -Repeat or return condition.
700 CALL VCHAR(OLDRMA            -Lines 700 and 710 plot the man.
    N,OLDCMAN,32)
710 CALL VCHAR(RMAN,CM
    AN,130)
720 GOTO 550                     -Loops back to line 550.
730 CALL SOUND(200,-7,0)         -Sound for hit condition.
735 SCORE = SCORE -10
740 FOR I = 1 TO 800             -Lines 740 and 750 are the timer.
750 NEXT I
760 GOTO 550
770 PRINT SCORE; "SCORE"         -Prints the score.
```

```
780 IF SCORE = 0 THEN 870
790 IF SCORE = -10 THEN
    910
800 IF SCORE = -20 THEN
    910
810 IF SCORE = -30 THEN
    960
820 IF SCORE = -40 THEN
    960
830 IF SCORE = -50 THEN
    960
840 FOR I = 1 TO 400
850 NEXT I
860 GOTO 450
870 PRINT "EXCELLENT"
880 FOR I = 1 TO 400
890 NEXT I
900 GOTO 450
910 PRINT "NOT BAD"
920 PRINT "BUT YOU CAN
    DO BETTER"
930 FOR I = 1 TO 400
940 NEXT I
950 GOTO 450
960 PRINT "CHECK YOUR EY
    ESIGHT AND TRY AGA
    IN"
970 FOR I = 1 TO 400
980 NEXT I
990 GOTO 450
```

-Lines 780 to 980 define the score conditions and prompts, and print the prompts.

-Loops back for another try.

When you run this game, the beginning prompts will appear and, then, 200 maze blocks will be displayed. You can increase or decrease the difficulty level of the Rat Trap by increasing or decreasing the number 200 in line 460.

When you press the cursor arrows, the little man will begin to move. It will take you a little practice, but you should be able to move through the maze without hitting any walls.

You can add your own touches to this game. You might want to hide treasures in the maze and score points for finding them. Or, you

could add a monster or two to chase you. Remember, keep your additions simple at first.

After working with the programs in this chapter, you should be able to build your own games. Work with them and have fun.

Color Codes

Color Codes

Color	Code No.	Color	Code No.
Transparent	1	Medium Red	9
Black	2	Light Red	10
Medium Green	3	Dark Yellow	11
Light Green	4	Light Yellow	12
Dark Blue	5	Dark Green	13
Light Blue	6	Magenta	14
Dark Red	7	Gray	15
Cyan	8	White	16

Character Codes

Standard Character Codes

Set 1		Set 2	
Code No.	Character	Code No.	Character
32	(space)	40	(
33	!	41	)
34	"	42	*
35	#	43	+
36	$	44	,
37	%	45	-
38	&	46	.
39	'	47	/

Set 3		Set 4	
Code No.	Character	Code No.	Character
48	0	56	8
49	1	57	9
50	2	58	:
51	3	59	;
52	4	60	<
53	5	61	=
54	6	62	>
55	7	63	?

Standard Character Codes (Cont'd)

Set 5		Set 6	
Code No.	**Character**	**Code No.**	**Character**
64	@	72	H
65	A	73	I
66	B	74	J
67	C	75	K
68	D	76	L
69	E	77	M
70	F	78	N
71	G	79	O

Set 7		Set 8	
Code No.	**Character**	**Code No.**	**Character**
80	P	88	X
81	Q	89	Y
82	R	90	Z
83	S	91	[
84	T	92	\
85	U	93	]
86	V	94	^
87	W	95	—

Unassigned Character Codes

Set 9	Set 10	Set 11	Set 12
96	104	112	120
97	105	113	121
98	106	114	122
99	107	115	123
100	108	116	124
101	109	117	125
102	110	118	126
103	111	119	127

Set 13	Set 14	Set 15	Set 16
128	136	144	152
129	137	145	153
130	138	146	154
131	139	147	155
132	140	148	156
133	141	149	157
134	142	150	158
135	143	151	159

Frequencies

Note Frequencies

Frequency	Note
110	A
117	A#.B
123	B
131	C (low C)
139	C#.D
147	D
156	D#.E
165	E
175	F
185	F#.G
196	G
208	G#.B
220	A (below middle C)
233	A#.B
247	B
262	C (middle C)
277	C#.D
294	D
311	D#.E
330	E
349	F
370	F#.G
392	G
415	G#.A
440	A (above middle C)
466	A#.B
494	B
523	C (high C)
554	C#.D
587	D
622	D#.E
659	E

Note Frequencies (Cont'd)

Frequency	Note
698	F
740	F#.G
784	G
831	G#.A
880	A (above high C)
932	A#.B
988	B
1047	C
1109	C#.D
1175	D
1245	D#.E
1319	E
1397	F
1480	F#.G
1568	G
1661	G#.A
1760	A

Index

TO THE READER

Sams Computer books cover Fundamentals — Programming — Interfacing — Technology written to meet the needs of computer engineers, professionals, scientists, technicians, students, educators, business owners, personal computerists and home hobbyists.

Our Tradition is to meet your needs and in so doing we invite you to tell us what your needs and interests are by completing the following:

1. I need books on the following topics:

_____

_____

_____

_____

_____

_____

2. I have the following Sams titles:

_____

_____

_____

_____

_____

_____

3. My occupation is:

_____ Scientist, Engineer
_____ Personal computerist
_____ Technician, Serviceman
_____ Educator
_____ Student
_____ D P Professional
_____ Business owner
_____ Computer store owner
_____ Home hobbyist
Other _____

_____

Name (print) _____

Address _____

City _____ State _____ Zip _____

Mail to: **Howard W. Sams & Co., Inc.**
Marketing Dept. #CBS1/80
4300 W. 62nd St., P.O. Box 7092
Indianapolis, Indiana 46206

22310

SAMS

The Tool Kit Series TI-99/4A Edition

To become computer literate and to keep your TI-99/4A computer purring along properly, you must have the correct tools. The "tools" in this book are short 5- to 15-minute subroutines that combine color, sound and graphics to form a variety of educational programs and computer games.

- Forget the structured method of learning programming. Forget commands, statements, and data structures.
- Forget those frustrating evenings with your User's Manual.
- Learn to look at programs in terms of their working parts—their subroutines—and learn how to write simple programs.
- Learn how to use color, sound and music, graphics, animation, and computational subroutines as "tools" to build programs.
- Discover the modular form of programming. Learn what each subroutine will do, how it can be changed, and what the variables control.

Don't spend long frustrating hours poring over programming books and manuals. Follow the "Tool Kit" approach to programming and learn to design your own games and quizzes. Keep your TI-99/4A computer humming happily.

Howard W. Sams & Co., Inc.
4300 West 62nd Street, Indianapolis, Indiana 46268 U.S.A.

$8.95/22310

ISBN: 0-672-22310-4